'De distributie van olie en gas'

Melvin van Dam

augustus 2003

'De distributie van olie en gas'

door
Melvin van Dam

Scriptie BedrijfsWiskunde en Informatica
Faculteit der Exacte Wetenschappen
Vrije Universiteit te Amsterdam

&

ORTEC bv
Business Unit Oil and Gas
Gouda

Begeleiders:

Elena Marchiori (VU)
Lambert van der Bruggen (ORTEC)
Niels Oudenaarde (ORTEC)

Augustus 2003

Voorwoord

Deze scriptie heb ik geschreven naar aanleiding van mijn afstudeerstage bij ORTEC bv te Gouda. Ik heb deze stage gelopen van februari 2003 tot en met augustus 2003 ter afronding van mijn opleiding BedrijfsWiskunde en Informatica (BWI) aan de Vrije Universiteit te Amsterdam.

ORTEC houdt zich onder andere veel bezig met logistiek, een vakgebied waar tijdens mijn studie mijn interesse voor gewekt is. Ik had dan ook een duidelijke voorkeur voor een afstudeerstage in die richting. Die mogelijkheid is mij door ORTEC geboden. Op de afdeling Olie en Gas van ORTEC is een programma genaamd ORION ontwikkeld. Dit is een ordergeneratiesysteem voor de secundaire distributie van benzine- en gasproducten. Dit wil zeggen de distributie vanuit opslagdepots naar tankstations en andere bedrijven zoals transporteurs of particulieren met een gastank.

In de huidige situatie vinden veel orders telefonisch plaats. Klanten van een oliemaatschappij (onder andere pomphouders, transporteurs en particulieren) bellen om een order door te geven. Een planner van de oliemaatschappij probeert binnengekomen orders zodanig te combineren tot ritten dat alle klanten op tijd beleverd worden.

Kort gezegd helpt ORION de planners van oliemaatschappijen de distributie zo in te richten dat ze van tevoren weten wanneer klanten in het systeem beleverd moeten worden. Zo is de kans heel klein dat die klanten zonder benzine of gas komen te zitten. Bovendien zijn de planners minder afhankelijk van de telefonische orders van de klanten. Een groot voordeel van dit systeem is dat planningen langer van tevoren bekend zijn, en zodoende de workload beter wordt verspreid over de week.

De stageopdracht bestaat uit het uitbreiden van de functionaliteit van ORION met:

- Nieuwe algoritmes voor het voorspellen van de vraag naar benzine of gas.
- Een module om klanten automatisch te classificeren of te clusteren.

Ik zal later de opdracht nog in detail toelichten.

Mijn dank gaat uit naar alle collega’s op de afdeling Olie en Gas voor de plezierige werksfeer. Ik heb de stageperiode mede door deze prettige sfeer met veel plezier doorlopen. In het bijzonder wil ik Niels Oudenaarde bedanken voor zijn hulp bij het inhoudelijke gedeelte van de stage en Lambert van der Bruggen voor zijn goede tips bij het schrijven van deze scriptie. Ook gaat een woord van dank uit naar Elena Marchiori voor het begeleiden van deze stage vanuit de VU.

Veel plezier bij het lezen van deze scriptie!

Samenvatting

De stageopdracht waaraan ik gewerkt heb betreft onder andere het analyseren van de huidige werking van ORION. Kort gezegd maakt ORION orders aan voor levering van olie- en gasproducten naar benzinestations. Het tijdstip en de grootte van de order worden berekend door op basis van de vraag uit het verleden een verwacht vraagprofiel op te zetten. Voor een uitgebreide beschrijving van de werking van ORION verwijs ik naar paragraaf 1.3 en hoofdstuk 2.

Een tweede onderdeel van de stageopdracht betreft het toevoegen van nieuwe voorspelalgoritmes aan ORION. Ik heb het 'seasonal decomposition' algoritme toegevoegd aan ORION. Dit algoritme houdt rekening met seizoenseffecten die optreden in de data. Voor de benzinemarkt levert dit geen voordeel op omdat hier geen seizoenseffecten voorkomen. Dit in tegenstelling tot de gasmarkt, die met zijn winterpiek een seizoensgebonden markt bij uitstek is. Het is daarom ook niet verwonderlijk dat de implementatie van dit algoritme heeft geleid tot betere voorspellingen voor veel klanten. In bijna de helft van de gevallen is de voorspelling met behulp van het 'seasonal decomposition' model beter geworden. Vaak is de verbetering fors.

Om de kwaliteit van de voorspellingen van algoritmes te vergelijken heb ik een module gemaakt die de 'performance' van een algoritme meet. De module vergelijkt werkelijke peilstanden van tanks uit het verleden met door ORION gegenereerde voorspellingen op basis van deze peilstanden. Voor een beschrijving van deze performance module verwijs ik naar paragraaf 5.2. Door met deze module een performance run uit te voeren wordt per klant bepaald welke voorspelmethodiek naar verwachting het beste presteert.

Tenslotte heb ik een module gemaakt die met gebruikmaking van de resultaten van een performance run de klanten classificeert. Door één druk op de knop kan voor een klantendatabase nu per klant het beste algoritme automatisch worden ingesteld. Deze module werkt goed, ondanks het feit dat het nog niet mogelijk is om automatisch verschillende parameter instellingen door te rekenen.

Inhoudsopgave

1 Inleiding

1.1 Opbouw Scriptie

In dit hoofdstuk zal ik een inleiding geven over ORTEC en ORION. Vervolgens zal ik in het tweede hoofdstuk de huidige werking van ORION in detail toelichten en aangeven op welke punten ORION verbeterd kan worden. In hoofdstuk 3 geef ik de precieze stageopdracht en de gekozen aanpak aan. Hoofdstuk 4 beschrijft de resultaten van het literatuuronderzoek wat ik voor deze stageopdracht uitgevoerd heb. In hoofdstuk 5 bespreek ik oplosmethoden die ik voor het probleem heb ontwikkeld. De testresultaten van de nieuwe modules staan in hoofdstuk 6, gevolgd door de conclusies en aanbevelingen in hoofdstuk 7 en 8. Tenslotte bevat hoofdstuk 9 een lijst van gebruikte literatuur en hoofdstuk 10 een aantal bijlagen. Zoals gezegd nu eerst een inleiding over ORTEC en ORION.

1.2 ORTEC

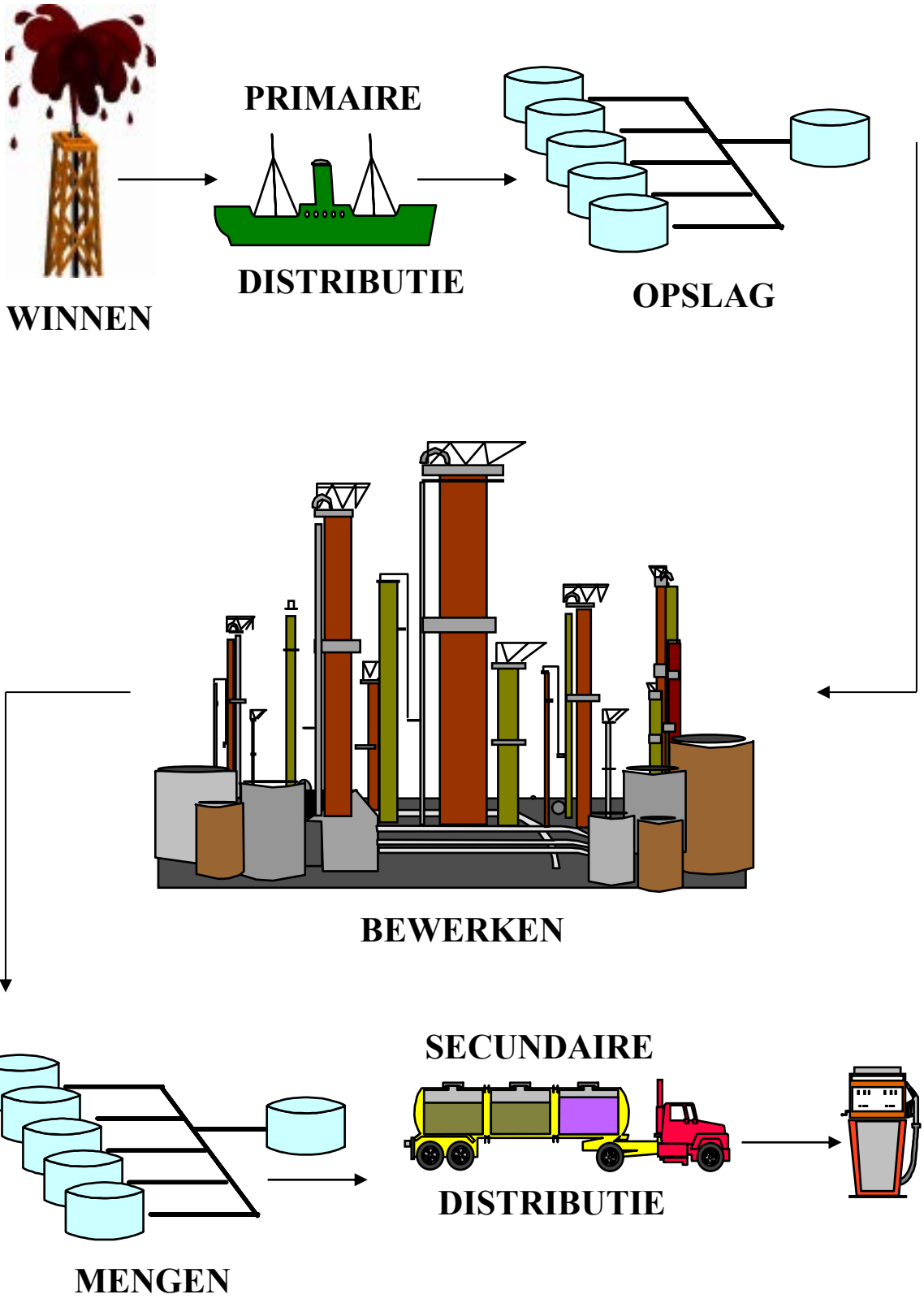
ORTEC bv is één van de grootste onafhankelijke advies- en softwarebedrijven op het gebied van Operations Research en Management Science. ORTEC biedt werk aan ongeveer 300 werknemers in verschillende vestigingen in binnen- en buitenland. Het hoofdkantoor staat in Gouda.

ORTEC is in twee opzichten een jong bedrijf. Ten eerste omdat het in 1981 is opgericht, maar bovenal omdat de gemiddelde ORTEC medewerker nog geen 30 jaar oud is. Er heerst binnen ORTEC een open en informele sfeer, en eigen initiatief wordt gestimuleerd en gewaardeerd.

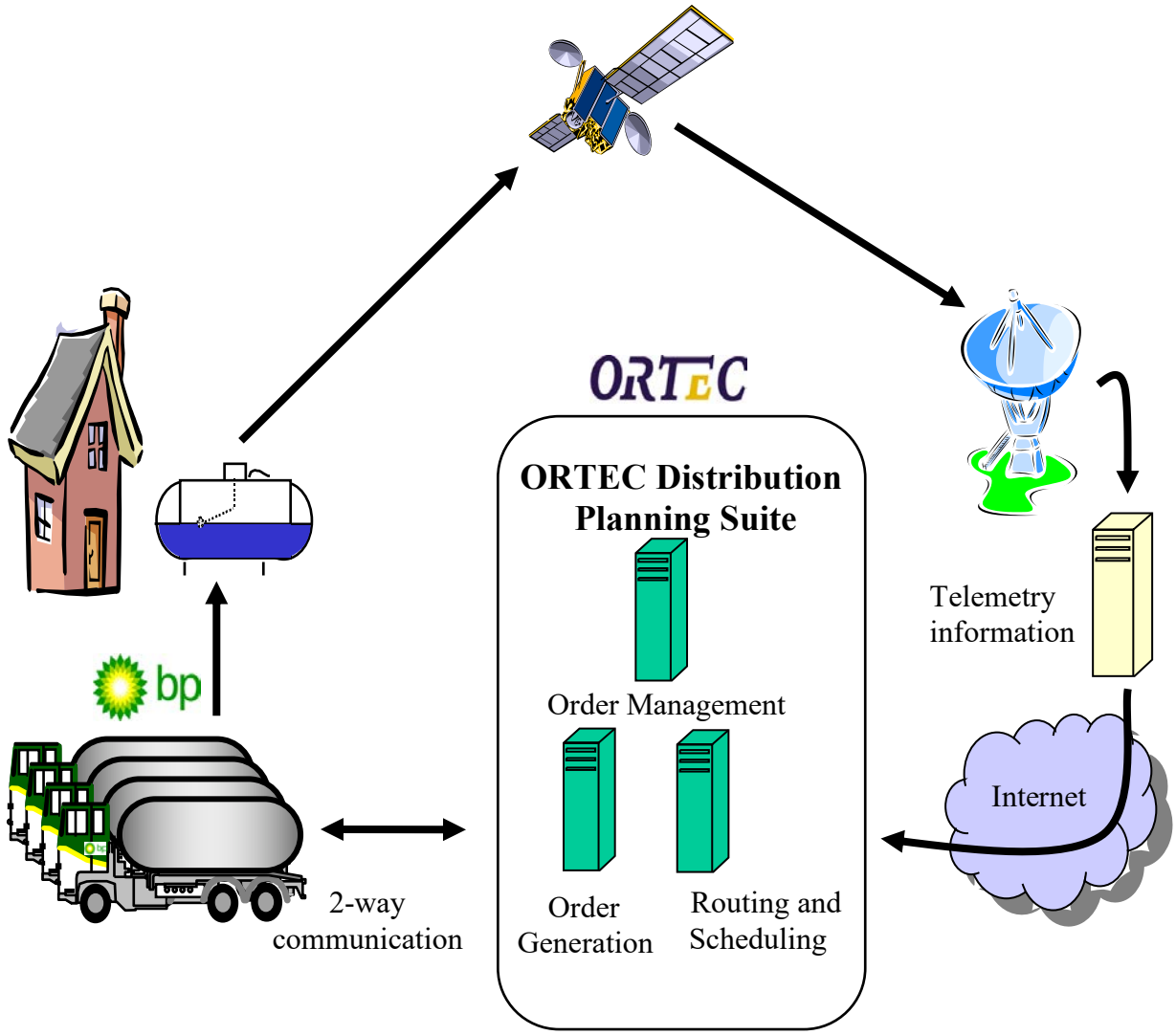
Het streven van ORTEC is om een leidende positie te bereiken op het gebied van geavanceerde planningsystemen. Daarbij willen ze zo breed mogelijk opereren.

De producten en diensten van ORTEC worden gekarakteriseerd door het gebruik van mathematische modellen en simulatie & optimalisatie technieken uit de Operations Research & Management Science. De gebieden waar ORTEC vooral actief is zijn Financieel en Logistiek. In de financiële poot van ORTEC bevinden zich onder andere de business units ‘Asset Liability Management’, ‘Performance Measurement’, ‘Financial Engineering’ en ‘Management Info Real Estate’. In de logistieke poot bevinden zich onder andere de business units ‘Transport & Distribution’, ‘Human Resource Management’, ‘Aviation & Traffic Management’, ‘Production Planning’ en ‘Oil & Gas’.

Binnen de ORTEC vestiging in Gouda ben ik werkzaam geweest bij de laatstgenoemde business unit: ‘Oil & Gas’. Deze business unit is geheel gericht op de supply chain van olie- en gasproducten. De supply chain voor benzine ziet er als volgt uit:



Voor de processen in deze supply chain zijn bij Oil & Gas een aantal planningsproducten ontwikkeld, onder andere voor de exploratie van ruwe olie, de productieplanning voor raffinaderijen, de primaire en secundaire distributie enzovoorts. Binnen Oil & Gas ben ik werkzaam geweest op de afdeling OGBS. Dit staat voor Oil and Gas Business Solutions. Binnen de afdeling OGBS ligt de focus op de secundaire distributie in de supply chain. Ter ondersteuning van deze secundaire distributie is onder andere ORION ontwikkeld. Dit is een voorraadbeheer- en ordergeneratiesysteem. Vaak wordt ORION bij een bedrijf geïmplementeerd in combinatie met SHORTREC. Dit is een ritplanningsysteem ontwikkeld door ORTEC. De orders die door ORION worden gegenereerd, worden dan in SHORTREC gecombineerd tot ritten en toegewezen aan trucks. ORION vormt samen met SHORTREC en administratie- en facturatiesystemen een totaalpakket voor de secundaire distributie van olie- en gasproducten. Dit pakket wordt de ORTEC Distribution Planning Suite genoemd. In het plaatje hieronder is de secundaire distributie van gas schematisch weergegeven, met de rol die de systemen van ORTEC daarin spelen.



1.3 ORION

Ik zal eerst even kort ingaan op wat ORION is en wat voor functionaliteit ORION de planners van de oliemaatschappijen kan bieden.

1.3.1 Het systeem ORION

ORION is zoals eerder genoemd een voorraadbeheer- en ordergeneratiesysteem, ontwikkeld voor oliemaatschappijen. ORION staat voor ORder and Inventory OptimizatioN. ORION wordt gebruikt voor de secundaire distributie van olie- en gasproducten naar benzinestations of gastanks vanuit grote depots. Omdat ORION nu nog voornamelijk voor benzineproducten gebruikt wordt, zal ik daar eerst op focussen.

Elk benzinestation heeft één of meerdere tanks waarin de benzines worden opgeslagen. Dat kan bijvoorbeeld voor diesel, euro95, euro98 en super elk één tank zijn. Maar het kan ook dat er meerdere tanks per product zijn. Het voorraadniveau in de tanks neemt af doordat mensen tanken. Eens in de zoveel tijd worden de tanks door een tankwagen bijgevuld. Voorheen ging het plannen van dit proces voornamelijk op basis van telefonische orders van pomphouders en ervaring van de planner. De planner heeft door zijn ervaring en het bijhouden van een kaartenbak enig inzicht in de afname van benzine bij de verschillende benzinestations. Een order bevat informatie over hoeveel er geleverd moet worden van welk product en wanneer.

Probleem echter is dat pomphouders vaak pas op het laatste moment bellen als ze bijna door hun voorraad heen zijn. De planner moet dan met spoed een tankwagen naar de pomphouder sturen, wat de oorspronkelijke planning vaak in de war schopt.

ORION is een systeem dat de besluitvorming van de planner kan overnemen en de telefoontjes van pomphouders overbodig maakt. ORION genereert automatisch orders voor de levering van benzines aan de stations. Men noemt dit een Vendor Managed Inventory Control systeem. Dat wil zeggen dat de leverancier het voorraadbeheer van de klant overneemt. Op basis van historische gegevens berekent ORION voor elke tank bij een station het verwachte toekomstige verbruik.

Als er meerdere tanks zijn voor één product wordt voor elke tank apart de toekomstige vraag voorspeld. Hier is een uitzondering op als de tanks onder de grond verbonden zijn. Dit worden ‘siphoned tanks’ genoemd. Benzine kan dan van de ene tank naar de andere stromen, zodat evenwicht ontstaat. In dit geval wordt de verwachte vraag wel voor beide tanks berekend, maar die is altijd in de verhouding van de tankinhoud. Als de ene tank 2x de inhoud heeft van de andere tank is zijn verwachte vraag dan ook 2x zo groot. Bij het voorspellen van het toekomstige verbruik wordt rekening gehouden met week- en dagprofielen. De afname van benzine zal namelijk niet elk uur van de dag en elke dag van de week gelijk zijn. Het voorspelde vraagprofiel is daarom per dag van de week verschillend.

Per station wordt vervolgens uitgerekend welke tank als eerste onder een bepaald voorraadniveau (de veiligheidsvoorraad) komt. Deze veiligheidsvoorraad wordt door de planner zelf ingevoerd, en dit gaat op basis van zijn ervaring en afspraken met de pomphouder. De redenen voor het aanhouden van een veiligheidsvoorraad zal ik in het volgende hoofdstuk aangeven. De tank waarbij de voorraad als eerste onder de veiligheidsvoorraad komt wordt de kritieke tank genoemd. Op basis van de laatst gemeten voorraad in de tank en met het verwachte toekomstige verbruik, wordt bepaald op welk tijdstip de tank uiterlijk beleverd moet worden. Dan wordt voor deze tank een order gegenereerd.

Indien mogelijk wordt er ook meteen geleverd in andere tanks bij hetzelfde station. Dit is om te voorkomen dat een tankstation meerdere keren per dag moet worden bezocht, of bijvoorbeeld twee dagen achter elkaar, voor een kleine levering van een product. Zo genereert ORION voor elk tankstation de orders. De planners kunnen overal in de planningen die door het systeem gegenereerd worden ingrijpen. Zo kunnen ze bijvoorbeeld wijzigingen aanbrengen in de ordergrootte die door ORION bepaald is, of in de datum of tijd waarop een order geleverd moet worden. Het is dus een beslissingsondersteunend systeem voor de planners van de oliemaatschappijen. Zolang een order nog niet geleverd is, zal de geplande order wel worden meegenomen in de berekeningen alsof deze geleverd is. Maar orders die alleen berekend zijn en nog niet geconfirmeerd, uitgevoerd of opgeslagen, worden weggegooid als er opnieuw een ordercalculatie wordt gedaan. Als een order daadwerkelijk is uitgevoerd, dan wordt de status van de geplande order aangepast en wordt de order als levering aan de historische gegevens toegevoegd, om de volgende keer te worden meegenomen in de berekeningen.

1.3.2 Proces van orders plannen

In deze paragraaf zal ik in het kort puntsgewijs de werkzaamheden van een planner schetsen. Deze paragraaf is niet noodzakelijk om de werking van ORION te begrijpen, maar geeft een aardig beeld hoe ORION in het dagelijkse proces van de planner wordt gebruikt.

• Nieuwe data import.	Als er nieuwe data is, bijvoorbeeld leveringen aan een benzinestation, wordt die geïmporteerd.
• Historie check.	De data wordt bekeken, eventuele uitschieters of verkeerde waarden worden aangepast.
• Orders berekenen in ORION.	ORION berekent op basis van de historie de nieuwe orders en een nieuw vraagprofiel.
• Logfile check.	Waarschuwingen die door ORION worden gegenereerd komen in een logfile te staan. Komt dit door fouten in de data dan wordt dit aangepast.
• Orders check.	De orders worden nagelopen, eventueel wordt er iets in gewijzigd.
• Export naar SHORTREC.	De berekende orders gaan naar SHORTREC.
• Schedule berekenen in SHORTREC.	SHORTREC berekent de ritten op basis van de orders.
• Import schedule in ORION.	De ritten worden geïmporteerd in ORION zodat de orders worden ge-update met de scheduled informatie.

2 Situatiebeschrijving

In dit hoofdstuk zal ik uitleggen hoe ORION werkt. Ik zal me in het bijzonder richten op de vraagvoorspelling en de ordergeneratie in ORION. Daarnaast zal ik aangeven welke verbeteringen in ORION nog mogelijk zijn.

2.1 Werking van de vraagvoorspelling in ORION

De vraagvoorspelling kan in de huidige situatie in ORION met twee verschillende algoritmes worden berekend, namelijk het 'moving average' algoritme en het 'exponential smoothing' algoritme. Eerst zal ik enkele begrippen toelichten die met de vraagvoorspelling te maken hebben. Daarna zal ik de twee algoritmes toelichten.

2.1.1 Begrippen voor de vraagvoorspelling

De begrippen die ik zal toelichten zijn: DIPs en DROPs, SALES, Absoluut verbruik, Dagprofiel, De 25-procentsregel, Openingstijden & speciale dagen en Weekprofiel.

2.1.1.a DIPs en DROPs

Een DIP is gedefinieerd als de meting van de hoeveelheid benzine die op een bepaald moment in een tank aanwezig is. Dat kan op verschillende manieren gebeuren, namelijk: met een peilstok, automatisch met behulp van telemetrie of door middel van het aflezen van een meter. Dit is per tank verschillend. Een DIP is dus de aanwezige hoeveelheid in een tank. De eenheid waarin gemeten wordt kan verschillen. In deze scriptie ga ik er vanuit dat de meet- en rekeneenheid liters is.

Een DROP is gedefinieerd als de aflevering of storting van een bepaalde hoeveelheid benzine in een tank. Over het algemeen is een DROP altijd gelijk aan de inhoud van één of meerdere compartimenten van een truck. Een situatie waarin dit niet het geval is, is bijvoorbeeld als een tank bij het vullen vol raakt voordat het compartiment leeg is.

Meestal is de chauffeur van de tankwagen degene die een DIP uitvoert. Daarom is er op de datum dat er een DROP plaatsvindt meestal ook een DIP, maar dit is niet noodzakelijk. Het is in het algemeen niet zo dat er elke dag bij een tankstation DIPs en/of DROPs plaatsvinden. Omdat deze metingen niet dagelijks plaatsvinden is het dagelijkse benzineverbruik per tank ook niet bekend. Een uitzondering hierop vormt het gebruik van telemetrie voor de DIPs. Dan kan wel dagelijks een automatische meting worden verricht. Maar telemetrie wordt nog weinig toegepast in verband met de relatief hoge kosten die hieraan zijn verbonden.

Om de vraagvoorspelling in ORION uit te voeren is het nodig om het gemiddelde benzineverbruik per dag van de week te weten. Omdat in het algemeen het exacte verbruik per dag niet bekend is, kan je ook niet zomaar het gemiddelde benzineverbruik per dag van de week uitrekenen. Hiervoor gebruikt ORION een toewijzing. Eerst wordt het verbruik tussen twee DIPs bepaald. Vervolgens wordt dit verbruik toegewezen aan de dagen van de week. Aan de hand van deze toewijzingen wordt de gemiddelde vraag per weekdag bepaald. Hoe dit proces in ORION precies werkt zal ik vanaf paragraaf 2.1.1c toelichten.

2.1.1.b SALES

Behalve DIPs en DROPs kan ORION ook overweg met verkoopgegevens, deze worden SALES genoemd. In dit geval is het benzineverbruik per tank per dag bekend op basis van verkoopinformatie. Nu is het niet nodig het verbruik toe te wijzen aan de dagen van de week zoals bij DIPs en DROPs het geval is. Het verbruik op een dag is exact gelijk aan de SALES voor die dag. En de voorraad in een tank is aan het eind van de dag precies de SALES hoeveelheid minder geworden (als er geen DROP was, anders komt deze afleverhoeveelheid er weer bij). Om de verwachte toekomstige vraag voor elke weekdag te berekenen kunnen de vraagvoorspelling algoritmes uit de 2.1.2 en 2.1.3 rechtstreeks gebruikt worden.

2.1.1.c Absoluut verbruik

Het absolute verbruik tussen twee achtereenvolgende DIPs op t_1 en t_2 wordt als volgt berekend:

$\forall i : t_1 < t_i < t_2 \qquad DIP_{t_1} + DROP_{t_i} - DIP_{t_2}$

Met andere woorden, het absolute verbruik tussen t_1 en t_2 is gelijk aan het voorraadniveau op t_1 plus alle leveringen tussen t_1 en t_2 verminderd met het voorraadniveau op t_2.

2.1.1.d Dagprofiel

Omdat de afname van benzine op een dag niet constant is, wordt rekening gehouden met een dagprofiel. Het dagprofiel geeft aan hoe de afname van de benzine over de dag verloopt. Voor elk benzinestation afzonderlijk kan per uur worden ingesteld hoeveel procent van de totale dagverkoop in dat uur valt. Deze profielen zijn handmatig in te stellen naar inzicht van de planner. Ik zal twee verschillende dagprofielen toelichten: het ‘Spits’profiel en het ‘Equal’profiel.

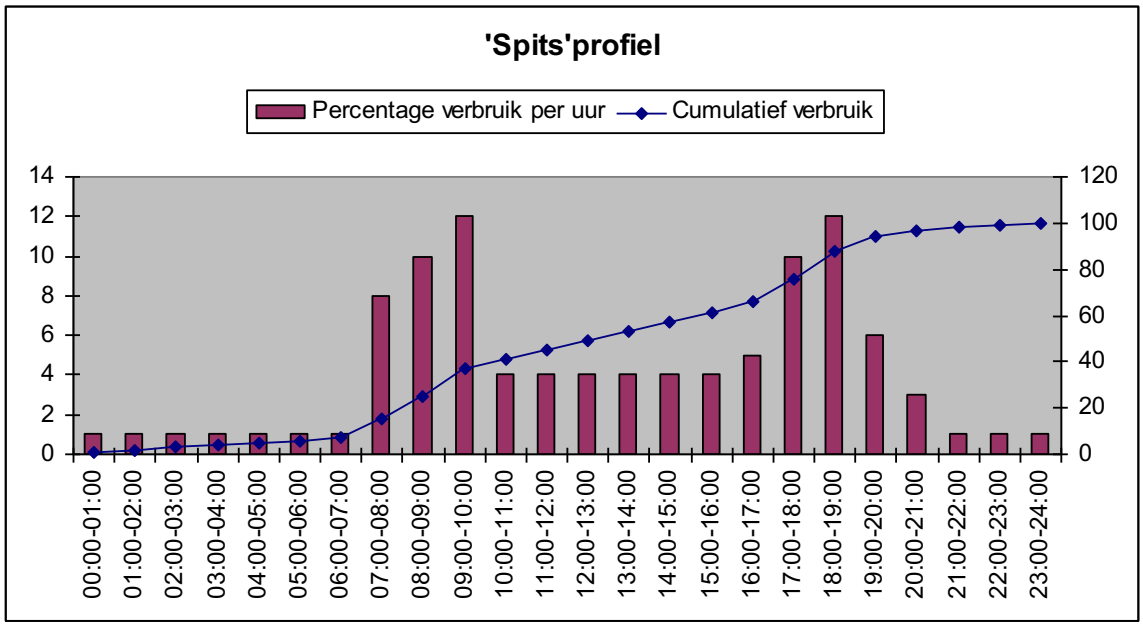
Als voor een benzinestation het ‘Spits’profiel wordt gebruikt, dan is de verkoop overdag groter dan tijdens de vroege ochtenduren en de late avonduren. Tijdens de spitsuren is er sprake van een behoorlijke piek. Zie ook Tabel 2.1 op de volgende pagina.

Tijd	Percentage	Tijd	Percentage
00:00-01:00	1	12:00-13:00	4
01:00-02:00	1	13:00-14:00	4
02:00-03:00	1	14:00-15:00	4
03:00-04:00	1	15:00-16:00	4
04:00-05:00	1	16:00-17:00	5
05:00-06:00	1	17:00-18:00	10
06:00-07:00	1	18:00-19:00	12
07:00-08:00	8	19:00-20:00	6
08:00-09:00	10	20:00-21:00	3
09:00-10:00	12	21:00-22:00	1
10:00-11:00	4	22:00-23:00	1
11:00-12:00	4	23:00-24:00	1

Tabel 2.1: Het ‘Spits’profiel

Uiteraard moet de verdeling over de dag sommeren tot 100 procent.

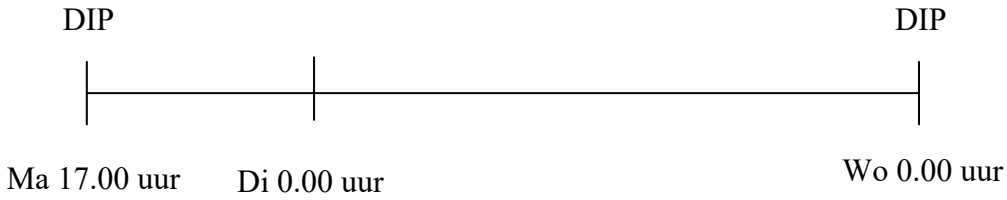
In Figuur 2.1 is te zien hoe de verdeling van de verkoop over de dag bij het ‘Spits’profiel verloopt.



Figuur 2.1: Grafiek van ‘Spits’profiel

Een ander mogelijk profiel is het ‘Equal’profiel. Bij dit profiel is de verkoop over de hele dag constant. De totale dagverkoop wordt gelijk over de hele dag uitgesmeerd. Per uur is het percentage dagverkoop gelijk aan 100/24 ≈ 4.167%.
In Bijlage A staan een tabel en grafiek van het ‘Equal’Profiel.

Het dagprofiel is van belang om de toewijzing van het verbruik aan de dagen van de week realistischer te maken. Omdat er zoals eerder gezegd alleen informatie over DIPs en DROPs is, moet het absolute verbruik tussen twee DIPs worden toegewezen aan de tussenliggende dagen. Stel nu dat de DIP op t_1 maandagmiddag om 17.00 uur plaatsvindt, en de DIP op t_2 woensdagochtend om 0.00 uur. Dan moet het absolute verbruik tussen deze DIPs worden toegewezen aan maandag en dinsdag.



Het is natuurlijk niet goed om te veronderstellen dat het verbruik op maandag en dinsdag allebei gelijk is aan ½ * absoluut verbruik. Op maandag heb je maar 7 uur gelegenheid gehad om benzine te verkopen (van 17.00 uur – 0.00 uur), terwijl je op dinsdag 24 uur had.

De totale openingstijd tussen de twee DIPs is 31 uur. Een andere methode is daarom om maandag 7/31 en dinsdag 24/31 van het absolute verbruik toe te wijzen. Deze methode is al beter, maar het kan nog beter. Het is in praktijk vaak zo dat bij benzinestations in de spitsuren veel meer verbruikt wordt dan de rest van de dag. Dit is bijvoorbeeld het geval bij stations langs snelwegen. Daar wil je in de voorspelling van het toekomstige verbruik ook rekening mee houden, om de planning zo goed mogelijk te maken. Vandaar de dagprofielen. In het voorbeeld gaat het bij gebruikmaking van het 'Spits'profiel dan als volgt:

17:00-18:00	10
18:00-19:00	12
19:00-20:00	6
20:00-21:00	3
21:00-22:00	1
22:00-23:00	1
23:00-24:00	1

Tabel 2.2: Fragment 'Spits'profiel

Cumulatief procentueel verbruik op maandag van 17.00 – 0.00 uur = 10 + 12 + 6 + 3 + 1 + 1 + 1 = 34% (Zie Tabel 2.2)

Het verbruik op dinsdag is 100% omdat de DIP op t_2 woensdagochtend 0.00 uur is. De toewijzing wordt dan:

maandag $\frac{34}{34+100}$ * absolute verbruik, dinsdag $\frac{100}{34+100}$ * absolute verbruik

In dit geval, waarbij de stations 24 uur per dag open zijn krijgen deze toewijzingen een gewicht gelijk aan het percentage dagverkoop. De toewijzing aan de maandag krijgt dus een gewicht van 0.34, die aan de dinsdag krijgt een gewicht van 1.

2.1.1.e De 25-procentsregel

De 25-procentsregel is ingevoerd om uitzonderlijke waarnemingen niet teveel invloed te laten hebben op de gemiddelde vraag per weekdag. Als een dag maar voor een deel meetelt wordt gekeken naar het percentage van de dagvraag die op dat dagdeel betrekking heeft. Bij het toewijzen, wordt een dag alleen meegenomen als minstens 25% van de dagvraag op het dagdeel betrekking heeft.

Voorbeeld:

00:00-01:00	1
01:00-02:00	1
02:00-03:00	1
03:00-04:00	1
04:00-05:00	1
05:00-06:00	1
06:00-07:00	1
07:00-08:00	8

Tabel 2.3: Fragment ‘Spits’profiel

Bij gebruik van het ‘Spits’profiel, is de vraag tussen 0.00 uur en 8.00 uur gelijk aan 1+1+1+1+1+1+1+8 = 15% < 25% van de totale dagvraag (zie Tabel 2.3).
In dit geval telt deze dinsdag niet mee voor het bepalen van de gemiddelde vraag over alle dinsdagen omdat aan de 25-procentsregel niet voldaan is.
In geval van het ‘Equal’profiel is wel aan de 25-procentsregel voldaan. Hier is de vraag tussen 0.00 uur en 8.00 uur gelijk aan 8 * 4.167 ≈ 33.33% > 25%. In dit geval zou deze dinsdag dus wel worden meegenomen in de berekening.

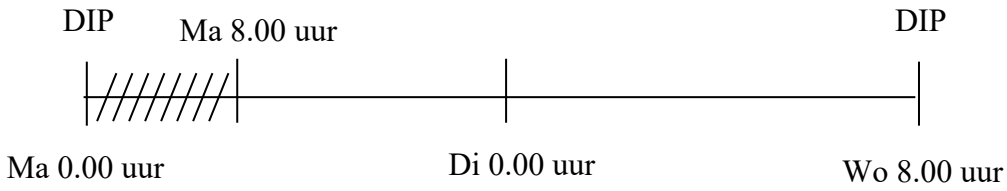
Metingen zijn nooit 100% nauwkeurig. Als 2 achtereenvolgende DIPs op een klein tijdsinterval plaatsvinden, bijvoorbeeld een uur, en je berekent de dagvraag, dan wordt die onnauwkeurigheid behoorlijk opgeblazen. Om dit te voorkomen is de 25-procentsregel ingevoerd. Bij drukbezochte stations kan het regelmatig voorkomen dat er twee DIPs snel achter elkaar plaatsvinden. Bijvoorbeeld als er op twee verschillende tijdstippen een andere tank wordt bijgevuld, maar wel alle tanks worden geDIPt.

NB. Puur theoretisch is de 25-procentsregel misschien niet nodig. De onnauwkeurigheid wordt wel opgeblazen, maar zo’n waarneming zal ook een lager gewicht krijgen dan waarnemingen over een hele dag, dus dat effect wordt ook wel weer wat teniet gedaan. Toch is het beter onnauwkeurige waarnemingen helemaal niet mee te nemen in plaats van ze met een laag gewicht mee te nemen. Daarom zit de 25-procentsregel toch in ORION.

2.1.1.f Openingstijden & speciale dagen

De gemiddelde vraag per dag van de week wordt gecorrigeerd voor de openingstijden van stations. De aanname in ORION is namelijk dat als een tankstation maar de helft van de dag open is, dat je dan ook de helft minder verkoopt dan wanneer het station de hele dag geopend is. Ik zal deze aanname toelichten.

Voorbeeld:



In het voorbeeld is het tankstation op maandag tot 8.00 uur gesloten (gearceerd), en er wordt gebruik gemaakt van het 'Spits'profiel.
Zoals in Tabel 2.3 op de vorige pagina is te zien, is het percentage dagverbruik tussen 0.00 uur en 8.00 uur gelijk aan 15%. Dus het percentage dagverbruik tussen 8.00 uur en 0.00 uur is dan 100 – 15 = 85%.

Maandag is het station 16/24 = 2/3 van de dag geopend. Dit is 66.67%.

Op maandag wordt nu verwacht dat het verbruik gelijk is aan 2/3 van het verbruik van dinsdag (omdat het station op maandag maar voor 2/3 geopend is). Bovendien wordt nog rekening gehouden met het dagprofiel. Dus 2/3 van 85% = 56.67%.

De toewijzing wordt dan:

maandag $\frac{56.67}{56.67+100}$ * absolute verbruik, dinsdag $\frac{100}{56.67+100}$ * absolute verbruik.

Het is een beetje de vraag in hoeverre dit een logische aanname is. Je zou kunnen zeggen dat je hier dan alleen met de dagprofielen zou moeten werken. Het is moeilijk te zeggen welke methode beter is. Stel dat een tankstation langs de snelweg staat en gebruik maakt van het 'Spits'profiel. Als dit station nu doordeweeks 24 uur per dag geopend is en in het weekend van 7.00 uur tot 20.00 uur, wat is dan de beste aanname? Het ligt niet voor de hand om te zeggen dat het tankstation in het weekend waarschijnlijk net zoveel verkoopt als op een doordeweekse dag tussen 7.00 uur en 20.00 uur. Je kan er van uitgaan dat bij een tankstation langs de snelweg op de doordeweekse dagen het meest verkocht wordt. Daarom is in ORION de aanname gemaakt dat de verkoop dan ook geschaald wordt naar de openingstijden.

In ORION kan de planner ook speciale dagen aangeven. Dat zijn dagen waarop de verwachte verkoop hoger of lager is dan normaal. Ik zal een paar voorbeelden geven.

Denk bij lagere verkoop bijvoorbeeld aan kerst waar de verwachte vraag naar benzine misschien lager is dan normaal. Dit zal vooral het geval zijn bij bijvoorbeeld tankstations in een industrie- of kantorengebied.
Een actueel voorbeeld is de crisis in Irak. Dit kan leiden tot een oliecrisis met als gevolg hele hoge olieprijzen. Dit kan de verwachte vraag naar benzine drukken.

Het is ook mogelijk dat er gedurende een bepaalde tijd een hogere verwachte vraag is. Bijvoorbeeld als het tankstation een aantal dagen het inmiddels beruchte ‘kwartje van Kok’ teruggeeft aan de pomp. Een ander voorbeeld is een aantal koopzondagen per jaar in een stad. De tankstations in de omgeving van de stad zullen deze koopzondagen waarschijnlijk een hogere vraag hebben dan gedurende een ‘normale’ zondag.

Voor deze dagen kan de planner het percentage voor de verwachte vraag aanpassen. Ook met het toewijzen van het verbruik aan de dagen van de week wordt hier rekening mee gehouden. Het verbruik wordt dus gecorrigeerd voor de speciale dagen, zodat deze dagen geen onterechte invloed hebben op de voorspelling voor de komende periode. Stel dat je bijvoorbeeld een dag 200% van de gebruikelijke vraag verwacht, dan telt deze dag maar voor 0.5 mee bij het berekenen van de gemiddelde vraag. Als de dag als een gewone dag zou meetellen, dan wordt het verbruik voor de toekomst waarschijnlijk te hoog ingeschat.

NB. De openingstijden, dagprofielen en speciale dagen kunnen per tankstation afzonderlijk worden ingesteld. Dit om per tankstation een realistische verwachte vraag te berekenen.

2.1.1.g Weekprofiel

Vanaf paragraaf 2.1.1c heb ik uitgelegd hoe de DIPs en DROPs door ORION worden omgezet in een gemiddelde vraag per dag van de week. Deze gemiddelden vormen het weekprofiel. Dit weekprofiel wordt gebruikt in de ordercalculatie, om te berekenen wanneer een tank naar verwachting op zijn veiligheidsvoorraad komt. In tegenstelling tot het dagprofiel, wat input is, volgt het weekprofiel uit de berekeningen in ORION.

2.1.2 Vraagvoorspelling met het 'moving average' algoritme

Zoals eerder opgemerkt kan de vraagvoorspelling in ORION op 2 manieren plaatsvinden. Eén daarvan is het 'moving average' algoritme.

Als input-parameter kan in ORION het aantal weken meegegeven worden waarover de historie moet worden meegenomen. Dit aantal noemen we n.
De 'moving average' methode ziet er in theorie als volgt uit:

$$voorspelling = \frac{\text{Som van absolute verbruik in de vorige n perioden}}{\text{n}}$$

Stel dat we nu op tijdstip t zitten, en de toekomstige vraag moet worden voorspeld. Dan wordt vanaf tijdstip t-n tot en met tijdstip t het verbruik tussen de DIPs toegewezen aan de dagen van de week, mits aan de 25-procentsregel wordt voldaan. Vervolgens neem je voor elke dag van de week het gewogen gemiddelde over de historie.

Voorbeeld:

In Tabel 2.4 staan van een bepaald product de toewijzingen in procenten van de dagvraag over de historie. DIPint staat voor DIP interval. Ik zal aan de hand van de 1e regel van de tabel uitleggen hoe de tabel moet worden gelezen.

DIPint = 1-2 →	Het gaat om het verbruik tussen DIP 1 en DIP 2.
Gem.ver = 12084 →	Het gemiddelde verbruik per dag tussen DIP 1 en DIP 2 was 12084 liter per dag. (Dit is berekend door het absolute verbruik te delen door het aantal dagen, rekening houdend met het tijdstip van de DIPs op de dag en een dagprofiel).
Ma = 16 →	Voor het berekenen van het gemiddelde verbruik op maandag telt 12084 liter voor 16% mee.
Di =100 →	Voor het berekenen van het gemiddelde verbruik op dinsdag telt 12084 liter voor 100% mee.

DIPint	Gem.ver. (l)	Ma	Di	Wo	Do	Vr	Za	Zo
1-2	12084	**16**	100	34				
2-3	9012			66	100	100	40	
3-4	11500	56					60	100
4-5	10234	44	100	**8**				
5-6	7954			92	100	100	27	

Tabel 2.4: Toewijzing in procenten van de dagvraag over de historie

Vetgedrukt zijn de percentages die kleiner zijn dan 25. Deze voldoen dus niet aan de 25-procentsregel en worden niet meegenomen.

Uitwerking voorbeeld:

Om de gemiddelde vraag voor maandag te berekenen wordt nu het gewogen gemiddelde genomen over de maandagen in de historie die voldoen aan de 25-procentsregel. Dat is in dit geval

$$\frac{(56*11500)+(10234*44)}{56+44} = 10943 \text{ (zie Tabel 2.4)}$$

Dus de gemiddelde vraag voor maandag is 10943 liter. Dit is tevens de schatting voor het verbruik op de eerstvolgende maandag. Zo worden alle dagen van de week berekend.

NB. Voor de gasmarkt is in ORION functionaliteit ontwikkeld om de voorspelling met het 'moving average' algoritme uit te breiden met een seizoenseffect. Dit kan handmatig door de planner worden ingesteld. Ik kom hier later nog op terug.

2.1.3 Vraagvoorspelling met het 'exponential smoothing' algoritme

Een andere manier van voorspellen is de 'exponential smoothing' methode.
In theorie ziet deze methode er als volgt uit:

$$F_t = \alpha Y_{t-1} + (1-\alpha)F_{t-1}$$

Met:
F_t = de voorspelling op tijdstip t
α = de smoothing constante ($\alpha \in [0,1]$)
Y_{t-1} = het echte verbruik op tijdstip t-1

Met andere woorden, de nieuwe voorspelling is gelijk aan α * het werkelijke verbruik in de vorige periode + (1-α) * de vorige voorspelling.
Dit impliceert dat alle historie wordt meegenomen, maar dat recentere waarnemingen zwaarder meetellen bij het bepalen van een nieuwe voorspelling.

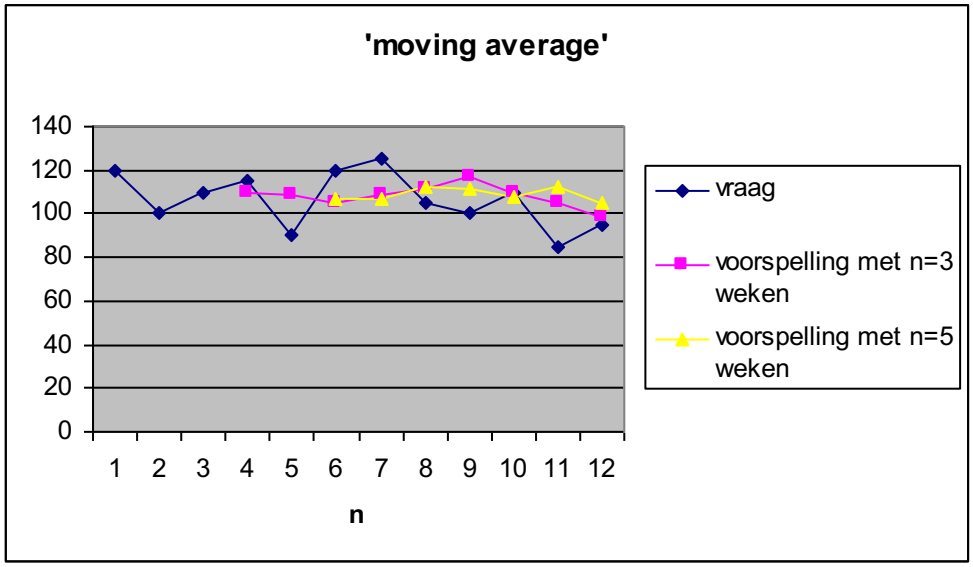
De eerste keer dat er een voorspelling gedaan wordt, vervangen we F_{t-1} door Y_{t-1} omdat er dan nog geen voorgaande voorspelling is. Er is voor gekozen α in ORION afhankelijk van het aantal dagen tussen de DIPs uit [0.5, 0.9] te kiezen. Hoe groter het aantal dagen tussen de DIPs hoe groter α. Op deze manier wordt er bij een groter aantal dagen tussen de DIPs minder waarde aan de vorige voorspelling gehangen, en meer aan het werkelijke verbruik. Dit is een logische keuze.

Het komt erop neer dat ORION, zodra er een nieuwe DIP is, het verbruik van de afgelopen periode berekent. Dit geeft in combinatie met de voorspelling van de afgelopen periode de nieuwe voorspelling.

In de volgende paragraaf zal ik de 'moving average' en 'exponential smoothing' methode vergelijken.

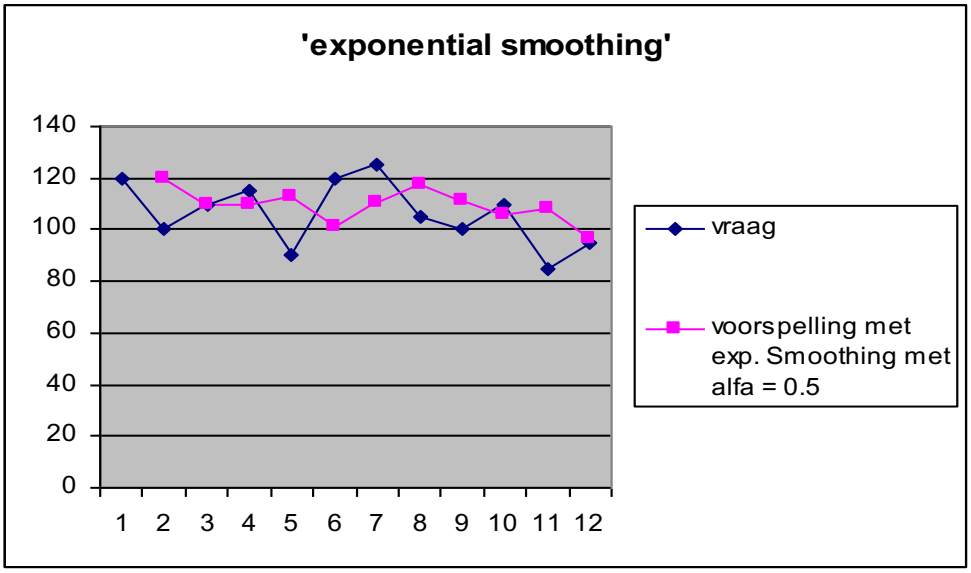
2.1.4 Vergelijking van ‘moving average’ en ‘exponential smoothing’

Het grote verschil tussen ‘moving average’ en ‘exponential smoothing’ is dat bij ‘exponential smoothing’ de waarnemingen worden gewogen. Bij het ‘moving average’ algoritme tellen alle waarnemingen even zwaar mee. Het idee achter het ‘moving average’ algoritme is dat wat in het verleden gebeurd is, in de toekomst weer zal gebeuren. Om te voorkomen dat uitschieters de voorspelling teveel beïnvloeden worden er n waarnemingen gemiddeld. Hoe groter n, hoe meer de voorspelling naar het gemiddelde toegaat. Dit betekent ook dat bij grote n de voorspelling niet zo snel zal reageren op veranderingen in de vraag. Dit is te zien in Figuur 2.2.



Figuur 2.2: Grafiek van ‘moving average’

Als je voor dezelfde data de voorspelling met het ‘exponential smoothing’ algoritme doet met $\alpha = 0.5$ dan krijg je de grafiek zoals in Figuur 2.3.



Figuur 2.3: Grafiek van ‘exponential smoothing’

Het is in Figuur 2.3 duidelijk te zien dat de voorspelling bij het ‘exponential smoothing’ algoritme schommelt. Dit in tegenstelling tot de voorspelling bij het ‘moving average’ algoritme waar de voorspelling redelijk constant is. Dit komt omdat bij het ‘expontial smoothing’ algoritme recente waarnemingen een zwaarder gewicht hebben. Dat betekent dat het algoritme eerder reageert op veranderingen in de vraag.

Bij een andere keuze voor α in het ‘exponential smoothing’ algoritme verandert de voorspelling.

- Bij een kleinere α (0 – 0.5) hangt de huidige voorspelling meer af van de vorige voorspelling en minder van het werkelijke verbruik.
- Bij een grotere α (0.5 – 1) hangt de huidige voorspelling meer af van het werkelijke verbruik en minder van de vorige voorspelling.

In de huidige versie van ORION moet de planner zelf aangeven met welke methode de vraagvoorspelling moet plaatsvinden. ORION geeft (nog) geen advies of voorstel om de planner met deze keuze te helpen. Als het verbruik redelijk constant is werkt het ‘moving average’ algoritme prima. Als er daarentegen regelmatig schommelingen over langere periodes in het verbruik zijn kan het ‘exponential smoothing’ algoritme beter werken, aangezien deze de schommelingen volgt. Dit kan gezien worden als een vuistregel om te bepalen wanneer welk algoritme gebruikt dient te worden.

NB. Zie bijlage B voor de dataset die ik heb gebruikt voor de vergelijking van deze twee algoritmes.

2.2 Werking van de ordergeneratie in ORION

Het genereren van orders om de planner te ondersteunen bij zijn taken is de voornaamste functionaliteit van ORION. Ik zal eerst weer een aantal begrippen toelichten die van belang zijn voor de ordergeneratie.

2.2.1 Begrippen voor de ordergeneratie

De begrippen die ik zal toelichten zijn: De veiligheidsvoorraad, Aantal dagen voorraad, Minimum en maximum DROP, De Round Off Quantity, Tankcapaciteit & maximum stock en Truckcapaciteit.

2.2.1.a De veiligheidsvoorraad

In het ideale geval vindt een DROP precies plaats op het moment dat een tank leeg raakt (het Just-In-Time principe). Dan is er geen overbodige voorraad aanwezig in de tank. Aan de pomp wordt ook geen ‘nee’ verkocht, omdat de tank tijdig weer gevuld wordt. Vanuit service oogpunt mag ‘nee’ verkoop ook niet plaatsvinden, er moet altijd benzine voorradig zijn. Een hoge voorraad aan houden is echter ook niet verstandig. Dit is dood kapitaal en kan voor een oliemaatschappij in de honderdduizenden liters lopen.

In praktijk is het onmogelijk precies te storten op het moment dat de tank leeg raakt. Dit heeft de volgende redenen:

- Intuïtief wil je orders kunnen combineren, dus meteen meerdere tanks bijvullen als je toch bij een station bent om een tank bij te vullen. Anders kan het voorkomen dat bij een station bijvoorbeeld 3 keer per dag een truck moet komen om een product te leveren. In dat geval is het beter om verschillende orders van een station te combineren tot een levering aan een station waarbij in meerdere tanks tegelijk wordt gestort. Er moet dan een afweging worden gemaakt tussen kosten van overbodige voorraad aanhouden en transportkosten.
- De truck die de order moet afleveren kan te laat komen.
- Het is over het algemeen zo dat de werkelijke vraag naar benzine afwijkt van de voorspelde vraag. Voorspellingen geven nooit zekerheid over de toekomst.

Om deze redenen wordt een veiligheidsvoorraad aangehouden.

NB. Het is in technisch opzicht vaak niet mogelijk om alle aanwezige voorraad in een tank te gebruiken. De tank kan dus niet tot de laatste liters leeg, de minimale voorraad die in een tank aanwezig moet zijn is de ‘bottomstock’. Uiteraard is de veiligheidsvoorraad altijd minimaal gelijk aan de ‘bottomstock’.

2.2.1.b Aantal dagen voorraad

Als ORION orders gaat genereren voor een benzinestation, wordt uitgegaan van de laatste DIPs in de tanks, en de verwachte toekomstige vraag voor elke dag van de week. Nu kan voor tank i berekend worden na hoeveel dagen de voorraad onder de gegeven veiligheidsvoorraad komt. Dit aantal dagen wordt NDAYS_i genoemd.

2.2.1.c Minimum en maximum DROP

De planner kan per tank een minimum en maximum afleverhoeveelheid invoeren. Dit wordt de minimum respectievelijk maximum DROP genoemd.

Een reden om een maximum DROP in te stellen is bijvoorbeeld dat er voor sommige producten een tank onder de grond ligt met een te grote capaciteit. Super bijvoorbeeld wordt tegenwoordig op veel plaatsen minder verkocht dan voorheen. Het kan voorkomen dat er dan nog een tank ligt met een capaciteit van 100.000 liter, en de weekvraag bijvoorbeeld 3000 liter is. Grote stortingen zijn dan niet zinvol, NDAYS_i wordt dan heel erg groot.

Een minimum DROP kan bijvoorbeeld zo worden ingesteld dat er altijd minstens een hoeveelheid ter grootte van een compartiment van een truck wordt geleverd. Het is zonde om een compartiment van 4000 liter maar met 1000 liter te vullen. Dan is de beladingsgraad van de truck te laag. (Uitzondering hierop is natuurlijk als een tank vol raakt voordat het compartiment leeg is, dan slaat de pomp af).

2.2.1.d De Round Off Quantity

De hoeveelheid benzine die geleverd wordt aan een tankstation wordt afgerond op een afrondhoeveelheid. Dit kan 1000, 500 of 100 liter zijn. Dit is de Round Off Quantity. Er zijn een aantal redenen waarom deze Round Off Quantity wordt gebruikt. Ten eerste is de afrondhoeveelheid de stapgrootte waarmee in de algoritmes van ordergeneratie wordt gerekend. Het scheelt behoorlijk in rekentijd om met een stapgrootte van 100, 500 of 1000 liter per keer de order op te hogen, in plaats van per liter. Bovendien is de orde van grootte van de dagverkoop van de meeste tankstations honderden of duizenden liters. Het heeft dus weinig toegevoegde waarde als ORION orders zou berekenen die op één liter nauwkeurig worden afgerond. Bovendien wordt vaak afgerond op volle compartimenten van een truck.

2.2.1.e Tankcapaciteit & maximum stock

Natuurlijk wordt er door ORION ook rekening gehouden met de maximale tankcapaciteit. Als er een order voor een station wordt gegenereerd mag de afleverhoeveelheid plus de verwachte aanwezige voorraad op dat moment niet boven de tankcapaciteit uitkomen. Er kan ook een maximum stock worden ingesteld. Dit kan zinvol zijn in het eerder genoemde voorbeeld van super. Als daarvoor een tank ligt met een veel te grote capaciteit, wil je die tank niet verder vullen dan een bepaald niveau. Dit wordt de maximum stock genoemd.

2.2.1.f Truckcapaciteit

Behalve tankcapaciteit heb je ook te maken met een truckcapaciteit. In ORION wordt per station een trucktype ingesteld waarmee dat station beleverd moet worden. Voor grote stations, met tanks met grote capaciteit, zal een grotere truck ingesteld worden dan voor kleinere stations. (Ervan uitgaande dat de oliemaatschappij met verschillende trucktypes rijdt). Sommige trucks zijn ook te groot om bij bepaalde stations te kunnen leveren. Vandaar dat per station een trucktype kan worden ingesteld. Bij het genereren van de orders in ORION wordt gerekend met de capaciteit van de ingestelde truck. Behalve een capaciteit in liters heeft elk type truck ook een gewichtsrestrictie. Door voor elk product van een order het aantal liters te vermenigvuldigen met het gewicht per liter van dat product wordt gekeken of aan deze gewichtsrestrictie is voldaan. Anders wordt de orderhoeveelheid aangepast. Omdat het standaard ingestelde truck-type niet altijd overeen zal stemmen met de door SHORTREC bepaalde truck, kan het nodig zijn dat de ordervolumes die gegenereerd zijn door ORION nog enigszins worden aangepast. Indien gewenst kan ORION al op compartimenten afronden.

2.2.2 Het 'Full Truck' algoritme voor ordergeneratie

In ORION zitten twee verschillende algoritmes voor ordergeneratie: 'Full Truck' en 'Equal Days'. Het 'Full Truck' algoritme, werkt als volgt:

Initialisatiestap:
Eerst wordt de tank met de kleinste $NDAYS_i$ bepaald (de kritieke tank). Dan wordt voor deze tank een order gegenereerd. De grootte van de order is de minimum DROP.

Algoritme:
- Bepaal opnieuw de tank met de kleinste $NDAYS_i$.
- Vergroot de order voor deze tank met de Round Off Quantity.

Stopcriterium: De truck is vol of alle tanks zijn vol.
- De order krijgt een kritieke datum en tijd mee. Dat is het tijdstip waarop de kritieke tank (waar de order voor gegenereerd is) naar verwachting leeg is.

Merk op dat het algoritme ervoor zorgt dat het zo lang mogelijk duurt voor er weer een tank leeg raakt. Mocht er tijdens de uitvoering van het algoritme een tank vol raken, dan gaat het algoritme verder met de tank die dan de kleinste $NDAYS_i$ heeft. Het doel van dit algoritme is zoveel mogelijk trucks vol te laten rijden. Dat betekent een hoge beladingsgraad en dus kostenbesparing.

2.2.3 Het 'Equal Days' algoritme voor ordergeneratie

Het 'Equal Days' algoritme werkt bijna hetzelfde als het 'Full Truck' algoritme. Het verschil doet zich voor wanneer er tijdens het algoritme een tank vol raakt. Bij het hierboven besproken 'Full Truck' algoritme, gaat het algoritme net zolang door met het ophogen van de order voor de andere tanks totdat alle tanks vol zijn of de truck vol is. Bij het 'Equal Days' algoritme daarentegen wordt de order voor de rest van de tanks dan zo aangepast dat de $NDAYS_i$ voor iedere tank even groot wordt, voor zover dit kan. Met andere woorden, de order die gegenereerd wordt met het 'Equal Days' algoritme vult de tanks zodanig bij dat na de DROP de verwachting is dat alle tanks bij die klant op hetzelfde moment weer leeg raken. Dit betekent dat de orderhoeveelheid niet altijd een volle truck is wat met het 'Full Truck' algoritme wel het geval is.

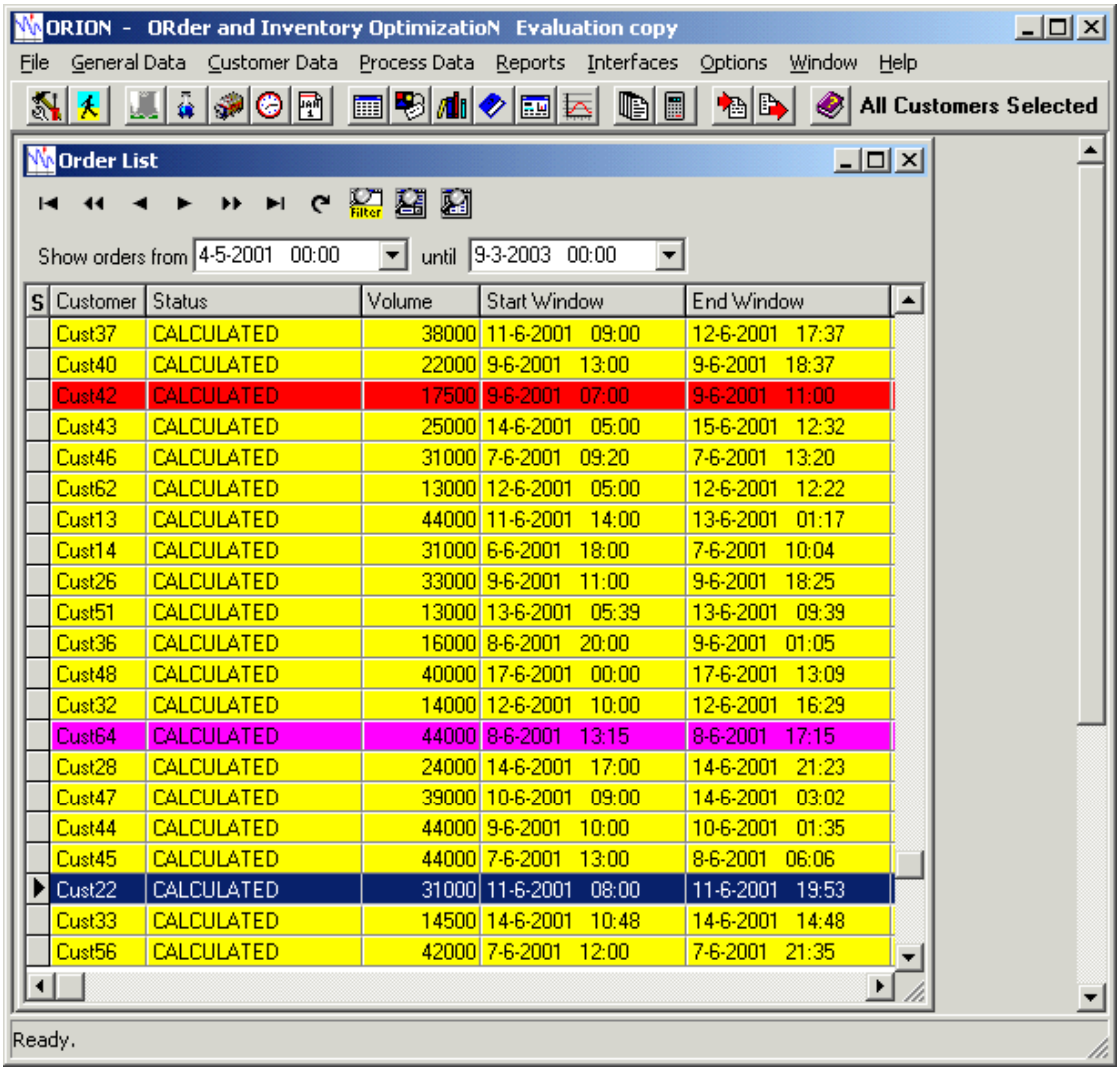
NB. Als er genoeg ruimte is in alle tanks, zodat er geen tank vol raakt, dan is de gegenereerde order door het 'Equal Days' algoritme gelijk aan de order gegenereerd door het 'Full Truck' algoritme.

Het nadeel van dit algoritme is dat het tot meer ritten kan leiden omdat niet alle trucks vol zijn bij vertrek. Een voordeel is dat er minder overbodige voorraad in de andere tanks van het station opgeslagen is, dus minder dood kapitaal. Als een order voor een station te klein is voor een volle truck, is een ander voordeel dat er makkelijker gecombineerd kan worden met orders voor andere stations.

Planners moeten zelf bepalen met welk algoritme de ordergeneratie wordt uitgevoerd.

2.2.4 Voorbeeld van gegenereerde orders

Voorbeeld van een door ORION gegenereerde orderlijst:



S	Customer	Status	Volume	Start Window	End Window
	Cust37	CALCULATED	38000	11-6-2001 09:00	12-6-2001 17:37
	Cust40	CALCULATED	22000	9-6-2001 13:00	9-6-2001 18:37
	Cust42	CALCULATED	17500	9-6-2001 07:00	9-6-2001 11:00
	Cust43	CALCULATED	25000	14-6-2001 05:00	15-6-2001 12:32
	Cust46	CALCULATED	31000	7-6-2001 09:20	7-6-2001 13:20
	Cust62	CALCULATED	13000	12-6-2001 05:00	12-6-2001 12:22
	Cust13	CALCULATED	44000	11-6-2001 14:00	13-6-2001 01:17
	Cust14	CALCULATED	31000	6-6-2001 18:00	7-6-2001 10:04
	Cust26	CALCULATED	33000	9-6-2001 11:00	9-6-2001 18:25
	Cust51	CALCULATED	13000	13-6-2001 05:39	13-6-2001 09:39
	Cust36	CALCULATED	16000	8-6-2001 20:00	9-6-2001 01:05
	Cust48	CALCULATED	40000	17-6-2001 00:00	17-6-2001 13:09
	Cust32	CALCULATED	14000	12-6-2001 10:00	12-6-2001 16:29
	Cust64	CALCULATED	44000	8-6-2001 13:15	8-6-2001 17:15
	Cust28	CALCULATED	24000	14-6-2001 17:00	14-6-2001 21:23
	Cust47	CALCULATED	39000	10-6-2001 09:00	14-6-2001 03:02
	Cust44	CALCULATED	44000	9-6-2001 10:00	10-6-2001 01:35
	Cust45	CALCULATED	44000	7-6-2001 13:00	8-6-2001 06:06
▶	Cust22	CALCULATED	31000	11-6-2001 08:00	11-6-2001 19:53
	Cust33	CALCULATED	14500	14-6-2001 10:48	14-6-2001 14:48
	Cust56	CALCULATED	42000	7-6-2001 12:00	7-6-2001 21:35

Figuur 2.4: Voorbeeld van een orderlijst in ORION

Per order is duidelijk aangegeven wat het totaalvolume is. Ook is een begintijd (‘start window’) en eindtijd (‘end window’) van de order gegeven. De eindtijd van de order is het moment waarop de order uiterlijk geleverd moet zijn. Dit is het moment waarop de kritieke tank de veiligheidsvoorraad bereikt. Hierbij wordt nog wel rekening gehouden met tijdstippen waarop levering mogelijk is. Als een tank op 19.15 uur kritiek wordt, maar het station mag niet na 18.00 uur beleverd worden, dan staat de eindtijd van de order op 18.00 uur. De begintijd van de order is het moment waarop de order voor het eerst in de tank past. (dus het moment waarop de voorraad plus de ordergrootte gelijk is aan de maximum voorraad). De begintijd en eindtijd worden samen het ‘delivery window’ genoemd. De minimale lengte van het delivery window kan worden ingesteld. Dit is standaard 4 uur. Als de delivery windows te kort zijn is er te weinig speling voor SHORTREC om toegestane ritten te berekenen. Als een gegenereerde order 4 uur voor de eindtijd van de order nog niet in de tank past worden de ordervolumes verlaagd zodat hier wel aan wordt voldaan.

De paars gekleurde orders geven een waarschuwing aan. Dit kan bijvoorbeeld zijn dat het voorraadniveau van de betreffende tank onder de veiligheidsvoorraad ligt, maar de tank nog niet leeg is. Als de voorraad in een tank naar verwachting 2 december aan het eind van de dag onder de veiligheidsvoorraad komt en je laat ORION 3 december de orders uitrekenen, dan zal de order paars gekleurd zijn. Deze situatie kan zich bijvoorbeeld voordoen als orders niet uitgeleverd worden. Als vanwege drukte niet alle orders voor een dag kunnen worden uitgevoerd, en een order wordt een dag opgeschoven, dan kan de voorraad in de tank de volgende dag onder de veiligheidsvoorraad liggen.

De rood gekleurde orders geven een fout aan. Dit kan bijvoorbeeld zijn dat de order veel te laat komt. In dit geval is de tank naar verwachting al leeg voor de nieuwe order arriveert.

2.3 Mogelijke verbeteringen in ORION

ORION is een geavanceerde planningstool die bij verschillende grote oliemaatschappijen geïmplementeerd is. Zo gebruikt BP het systeem wereldwijd in meer dan 10 landen, Q8 in de benelux, TotalFinaElf in België, Galp in Portugal en onder andere GULF in Nederland. Deze oliemaatschappijen ondervinden flinke voordelen na de implementatie van ORION in combinatie met het ritplanningsysteem SHORTREC. Het blijkt bijvoorbeeld dat het volume van de DROPs per tank behoorlijk omhoog gaat. Dit betekent minder ritten, en dus minder kosten. Ook worden pieken in de vraag uitgesmeerd. Dit is te verklaren door het feit dat ORION verder vooruit kijkt dan de pomphouders. Als pomphouders zelf bellen om een order door te geven dan moet deze order over het algemeen snel geleverd worden, omdat een pomphouder pas belt als de tanks bijna leeg raken. Omdat ORION verder in de toekomst kijkt kunnen er efficiëntere route’s gereden worden. Er is meer flexibiliteit in het schema zodat de workload beter kan worden uitgesmeerd over een week. Ook de weekendpiek kan worden opgevangen. Dit betekent dat minder vaak trucks op zaterdag ingezet hoeven te worden, wat relatief duur is. Een gevolg van deze zogenaamde ‘workload-smoothening’ is dat oliemaatschappijen vaak met minder trucks toe kunnen.

Toch zijn er nog situaties aan te geven waarin de huidige manier van voorspellen in ORION tekortschiet of verbeterd kan worden. In paragraaf 2.3.1 en 2.3.2 zal ik voor de gasmarkt respectievelijk benzinemarkt aangeven welke verbeteringen mogelijk zijn.

2.3.1 Verbeteringen voor de gasmarkt

In het vervolg zal ik de term gasklanten zo nu en dan gebruiken. Ter verduidelijking, onder gasklanten worden in deze scriptie onder andere verstaan: particulieren die gas gebruiken om te koken en het huis te verwarmen en bedrijven die gas gebruiken voor hun bedrijfsvoering. In Nederland is bijna iedereen aangesloten op het gasnet maar in andere landen (bijvoorbeeld Frankrijk of Schotland) hebben veel particulieren in afgelegen gebieden een gastank.

ORTEC wil met ORION haar positie op de gasmarkt gaan versterken. De gasmarkt zit echter heel anders in elkaar dan de benzinemarkt.

Omdat die gasmarkt zo anders is werkt ORION voor de gasmarkt nog niet goed genoeg. De volgende problemen doen zich voor:

- De leveringen aan gasklanten vinden meestal veel minder frequent plaats. Soms zijn er maar 2 of 3 DIPs per jaar. Er is dus veel minder data beschikbaar, en dat maakt het moeilijk om betrouwbare voorspellingen te doen.
- De afname van gas is sterk aan seizoenseffecten onderhevig. In de winter is de afname vaak vele malen groter dan in de zomer. Dan is de vraag naar gas ineens zo groot dat het voor de leveranciers bijna niet mogelijk is om het bij te benen. Nu wordt in ORION nog helemaal geen rekening gehouden met seizoenseffecten.

Daarom werkt ORION voor gas minder goed. Het is goed denkbaar dat de voorspelling een stuk beter kan worden bij het gebruik van een voorspelmethodiek waarin rekening wordt gehouden met een seizoenseffect. Het is daarom de bedoeling om naast het ‘moving average’ en het ‘exponential smoothing’ algoritme een nieuwe voorspelmethodiek in ORION te implementeren. Merk hierbij wel op dat elke voorspelmethodiek afhankelijk is van de aangeleverde data. Zowel in kwaliteit als kwantiteit geldt deze afhankelijkheid.

- Kwantiteit: Voor een gastank waar maar 2 of 3 DIPs per jaar plaatsvinden kan je geen goede voorspelling genereren als je verder geen informatie hebt. Maar dit betekent natuurlijk niet dat de voorspelling niet beter kan worden door gebruik van een andere voorspelmethodiek.
- Kwaliteit: Als DIP en DROP data onnauwkeurig zijn, zullen voorspellingen omtrent verbruik ook niet goed zijn. (Het ‘garbage in, garbage out’ principe).

Een ander verschil tussen de gasmarkt en de benzinemarkt is dat in de gasmarkt een klantendatabase uit duizenden klanten bestaat, en in de benzinemarkt vaak maar uit enkele honderden. De planner kent zijn klanten in de gasmarkt daarom veel minder goed als in de benzinemarkt. Omdat het veel tijd kost om voor al die klanten de juiste voorspelmethodiek met de juiste parameters te bepalen, moet daar ook een andere oplossing voor komen. Het is de bedoeling dat er een module in ORION komt die klanten automatisch kan classificeren of clusteren.

2.3.2 Verbeteringen voor de benzinemarkt

Voor de benzinemarkt is ORION een goed werkend programma. Bij de meeste benzinestations vinden zeer regelmatig DIPs en DROPs plaats, dus is er voldoende data om redelijk betrouwbare voorspellingen voor de vraag te doen.

Wat wel beter kan is de classificering die ik in paragraaf 2.3.1 al genoemd heb. Nu kan de planner zelf de stations aan een klasse toevoegen, maar dat gaat op basis van inzicht van de planner. Deze klassen bevatten eigenschappen als minimum drop, Round Off Quantity en voorspelalgoritme. De voorspelmethodiek waarmee gerekend wordt hangt dus af van de klasse waarin de planner een station stopt. Over het algemeen heeft een planner er waarschijnlijk geen zicht op welk voorspelalgoritme voor welk station goed werkt. Het zou dus mooi zijn als er een module komt waarmee stations automatisch worden geclassificeerd of geclusterd.

3 De stageopdracht

Nu ik de werking van ORION heb toegelicht zal ik de opdracht en de aanpak die ik heb gekozen laten zien.

3.1 Inhoud opdracht

- Analyseer de huidige werking van ORION.
- Onderzoek andere forecasting algoritmes.
- Onderzoek methodes om klanten te classificeren of te clusteren.
- Ontwikkel en implementeer nieuwe forecasting algoritmes in ORION.
- Ontwikkel en implementeer een module in ORION die alle klanten in de database automatisch classificeert of clustert.
- Genereer resultaten op real-life data en maak een vergelijking van de verschillende methoden.

3.2 Aanpak

Het eerste punt van de opdracht, het analyseren van de huidige werking heb ik gedaan door veel met ORION te werken, de handleiding goed door te lezen en vragen te stellen aan mijn begeleiders binnen ORTEC. Voor het uitvoeren van de andere onderzoeksvragen heb ik een literatuuronderzoek uitgevoerd met nadruk op forecasting algoritmes en het classificeren/clusteren. Bij dit onderzoek heb ik gebruik gemaakt van het internet, mijn begeleidster op de VU en de universiteitsbibliotheek. Vervolgens heb ik de resultaten van het literatuuronderzoek gebruikt om een oplosmethode te ontwerpen voor de ORION problematiek. Dit heeft geresulteerd in modules die ik ontwikkeld en geïmplementeerd heb in ORION. Uiteindelijk heb ik de nieuwe modules getest op een dataset uit de praktijk. Aan de hand van een aantal criteria heb ik de prestaties van de verschillende algoritmes beoordeeld en de resultaten vergeleken met de huidige methodes.

In het volgende hoofdstuk volgen de resultaten van het literatuuronderzoek.

4 Literatuuronderzoek

Ik heb literatuuronderzoek gedaan naar voorspelmethodieken waarin rekening wordt gehouden met seizoensinvloeden. Er zijn een aantal van deze methodieken. Een voorbeeld van zo'n methodiek is het Holt-Winters model. Voor uitleg over dit model verwijs ik naar [6]. Een ander voorbeeld is een autoregressiemodel. Hiervoor verwijs ik naar [4]. Ik heb echter gekozen voor het 'seasonal decomposition' model. Ik heb voor dit model gekozen omdat het veel gebruikt is en relatief eenvoudig te implementeren in de ontwikkelomgeving van ORION. In paragraaf 4.1 zal ik het 'seasonal decomposition ' model beschrijven.

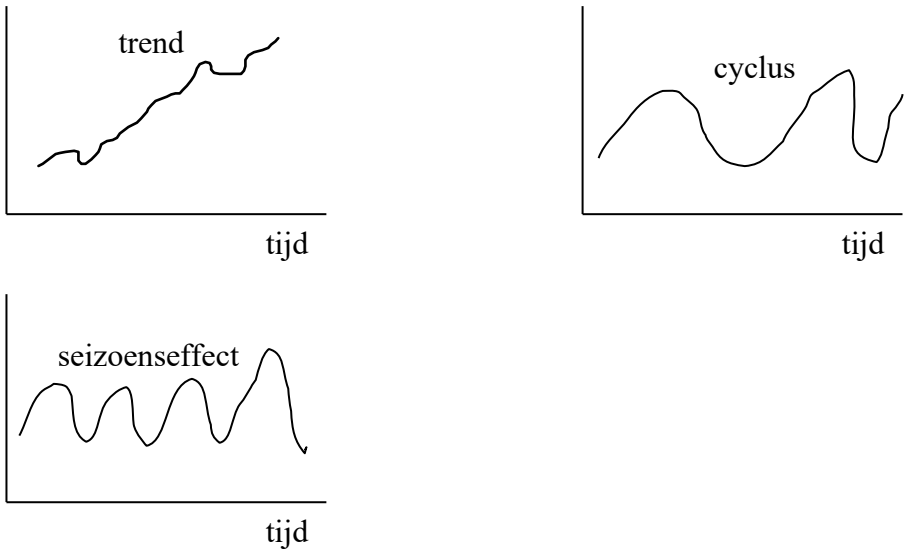
Er bestaat in ORION al functionaliteit om de vraagvoorspelling met het 'moving average' algoritme uit te breiden met een seizoensprofiel. In dit geval stelt een gebruiker zelf voor een aantal verschillende periodes per jaar het percentage van het jaarverbruik in. Om zelf voor een dataset een seizoensprofiel op te stellen heb ik gebruik gemaakt van de 'graaddagen' methode. Deze methode beschrijf ik in paragraaf 4.3.

4.1 'Seasonal decomposition', een voorspelalgoritme gebaseerd op seizoenen

De data in de vorm van DIPs en DROPs en/of SALES kan worden beschouwd als een tijdreeks. Een tijdreeks is opgebouwd uit de volgende elementen:

- Trend → een op of neergaande beweging op lange termijn.
- Seizoenseffect → een periodiek patroon, herhaalt zich elk jaar.
- Cyclus → een golfbeweging rond de trend, ook wel conjunctuur genoemd.
- Random variations → toevallige bewegingen die geen patroon volgen.

Ter illustratie staan in Figuur 4.1 de componenten van een tijdreeks afgebeeld.



Figuur 4.1: Componenten van een tijdreeks

Een tijdreeks kan als volgt worden gemodelleerd (‘seasonal decomposition’ model):

$$y = trend \times seizoenseffect \times cyclus \pm random\ variations$$

De random variations zijn niet te voorspellen. Een cyclus is heel moeilijk uit de data te halen of te voorspellen. Daarom wordt in het ‘seasonal decomposition’ model de aanname gemaakt dat de cyclus wordt opgevangen door de trend. Het model wat overblijft is dan:

$$y = trend \times seizoenseffect$$

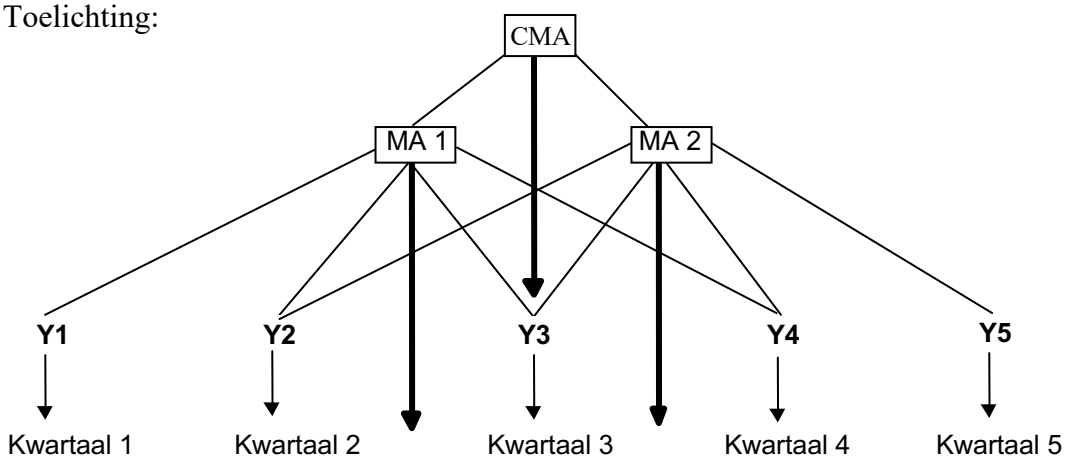
In paragraaf 4.1.1 en 4.1.2 zal ik achtereenvolgens de berekening van het seizoenseffect en de trend in het model toelichten.

4.1.1 Schatten van het seizoenseffect

Het seizoenseffect moet worden geschat uit de data. Eerst geef ik het stappenplan van de methode die daarvoor wordt gebruikt, vervolgens zal ik de stappen toelichten. Ik zal er in de toelichting vanuit gaan dat er met kwartalen gerekend wordt.

Stap 1: Bereken MA, de moving averages over een jaar.
Stap 2: Bereken CMA, de gecentreerde moving averages.
Stap 3: Bereken de seizoensratio's.
Stap 4: Bereken de seizoensindices.
Stap 5: Schaal de seizoensindices.

Toelichting:



Ad 1: In bovenstaand schema is te zien dat de moving averages worden genomen over 4 opeenvolgende kwartalen. MA1 is het gemiddelde van de som van kwartaal 1 tot en met 4, MA2 is het gemiddelde van de som van kwartaal 2 tot en met 5.

Ad 2: De moving averages worden gecentreerd omdat een jaar een even aantal kwartalen heeft. Hierdoor vallen de moving averages precies tussen de kwartalen in zoals te zien is in bovenstaand schema. Door het gemiddelde van 2 opeenvolgende moving averages te nemen (bijvoorbeeld van MA1 en MA2) wordt dit probleem opgelost. CMA is dus de gecentreerde moving average behorende bij kwartaal 3.

Ad 3: Bereken de seizoensratio's door het verbruik in een kwartaal te delen door de gecentreerde moving average behorende bij dat kwartaal.

Ad 4: Bepaal de seizoensindices door voor elk kwartaal de seizoensratio's behorende bij dat kwartaal op te tellen en hier het gemiddelde over te nemen.

Ad 5: Schaal de seizoensindices zodanig dat het gemiddelde van de seizoensindices 1 is. Dit om het effect van eventuele afrondfouten ongedaan te maken.

De Tabellen 4.1 en 4.2 geven een schematische weergave van stap 1 t/m 4 van bovenstaande methode.

Uitleg Tabel 4.1

Nr → volgnummer van de metingen.
Kw → kwartaal waarin de meting plaatsvond.
Ver → het verbruik behorende bij de meting.

Nr	Kw	Ver	Moving Average (stap 1)	Gecentreerd MA (stap 2)	Ratio's (stap 3)
1	1	Y1			
2	2	Y2	MA1 = (Y1 + Y2 + Y3 + Y4) / 4		
3	3	Y3	MA2 = (Y2 + Y3 + Y4 + Y5) / 4	CMA1= (MA1 + MA2) / 2	R1 = Y3/CMA1
4	4	Y4	MA3 = (Y3 + Y4 + Y5 + Y6) / 4	CMA2= (MA2 + MA3) / 2	R2 = Y3/CMA1
5	1	Y5	MA4 = (Y4 + Y5 + Y6 + Y7) / 4	CMA3= (MA3 + MA4) / 2	R3 = Y3/CMA1
6	2	Y6	MA5 = (Y5 + Y6 + Y7 + Y8) / 4	CMA4= (MA5 + MA6) / 2	R4 = Y3/CMA1
7	3	Y7			
8	4	Y8			

Tabel 4.1: Stap 1 t/m 3 voor het schatten van het seizoenseffect

In het voorbeeld is er maar 2 jaar data (8 kwartalen). Dat levert voor elk seizoen precies 1 seizoensratio op. In praktijk is het beter als er meer data beschikbaar is. Dan worden de seizoensratio's gemiddeld. In tabel 4.2 staat (Ri) voor de seizoensratio's die berekend worden als er meer data beschikbaar is. Si staat voor seizoensindex

Jaar	1e kwartaal	2e kwartaal	3e kwartaal	4e kwartaal
1	-	-	R1	R2
2	R3	R4	(R5)	(R6)
3	(R7)	(R8)	(R9)	(R10)
4	..	..	..	..
Si	Si1 = gemiddelde van R3 + (R7) + ..	Si2 = gemiddelde van R4 + (R8) + ..	Si3 = gemiddelde van R5 + (R9) + ..	Si4= gemiddelde van R6 + (R10) + ..

Tabel 4.2: Stap 4 voor het schatten van het seizoenseffect

4.1.2 Schatten van de trend

Na het schatten van het seizoenseffect wordt de trend bepaald.

Stap 1: Verwijder het seizoenseffect uit de data.
Stap 2: Gebruik de least-squares methode om een trendlijn te krijgen die op de data fit.
Stap 3: Bereken de trendwaarde voor elke periode.

Ad 1: Deel het verbruik in een kwartaal door de seizoensindex behorende bij dat kwartaal.

Ad 2: Zoek de lijn $Y = aX + b$ waarvoor $\sum_{i=1}^{n}(Y_i - a - bX_i)^2$ minimaal is.

Als de data na verwijdering van het seizoenseffect stabiel is, en er dus geen sprake is van een trend, dan worden stap 2 en 3 niet uitgevoerd. Er wordt dan geen trend berekend.

Of de data al dan niet stabiel is kan bijvoorbeeld grafisch worden bekeken met een simpele plot van de data. Als er een rechte lijn door de datapunten getrokken kan worden met een richtingscoëfficient van 0, dan is de data stabiel. Het kan ook met een lineaire regressie op de data.

4.1.3 Berekenen van de voorspelling

De voorspelling wordt berekend met:

$$F_t = trend_t \times seizoensindex_t$$

Met andere woorden, de voorspelling op tijstip t is de trend op tijdstip t vermenigvuldigd met de seizoensindex op tijdstip t.

Zoals eerder vermeld wordt geen trend berekend als de data na het verwijderen van seizoenseffecten stabiel is. In dat geval wordt het ‘exponential smoothing’ algoritme toegepast op de data waar het seizoenseffect uit verwijderd is. Dit levert een verwachte vraag op voor de komende periode. Vermenigvuldigen van deze verwachte vraag met de juiste seizoensindex levert de voorspelling op.

4.2 Classificering / Clustering

Classificeren is het rangschikken in klassen. Clusteren is het samenbrengen in groepen met gelijke eigenschappen. Het grootste verschil tussen classificeren en clusteren is, dat bij classificeren de klassen van te voren bekend zijn en bij clusteren niet.

De bedoeling is klanten automatisch te kunnen toewijzen aan één van de vraagvoorspellingsalgoritmes. De klassen zijn dus al bekend, namelijk ‘moving average’, ‘exponential smoothing’ en ‘seasonal decomposition’.

Daarom is het voldoende om alleen naar classificatie te kijken. Ik heb voor de classificatie gebruik gemaakt van een zelf ontwikkelde ‘performance’ module. Deze module heb ik ontwikkeld om de prestaties van de voorspelmethodieken te vergelijken. Deze module bleek ook geschikt om te classificeren. Voor een uitleg van deze ‘performance’ module verwijs ik naar paragraaf 5.2.

4.3 Graaddagen

Het gasverbruik van een huishouden is sterk afhankelijk van de buitentemperatuur. Om het gasverbruik te corrigeren voor de temperatuur is de graaddagen methode een veel gebruikt hulpmiddel. Informatie over graaddagen is te vinden in [5].

De graaddagen methode werkt als volgt:

$$graaddagen = 0, \quad als\ gemiddelde\ etmaaltemperatuur \geq 14^{o}\,\mathrm{C}$$
$$graaddagen = referentietemperatuur - gemiddelde\ etmaaltemperatuur, \quad anders$$

Het is gebruikelijk een referentietemperatuur van 18 ºC aan te houden. Het aantal graaddagen is altijd groter of gelijk aan 0. Als de gemiddelde etmaaltemperatuur boven de 14 ºC ligt, dan is het aantal graaddagen gelijk aan 0.

Voorbeeld:

Bij een gemiddelde etmaaltemperatuur van 0 ºC is het aantal graaddagen 18.
Bij een gemiddelde etmaaltemperatuur van 9 ºC is het aantal graaddagen 9.

Volgens de graaddagen methode is het verschil in gasverbruik bij een gemiddelde etmaaltemperatuur van 0 ºC en 9 ºC dus een factor 2.

5 Oplosmethoden

5.1 Oplosmethoden voor ‘seasonal decomposition’

De in het vorige hoofdstuk geschetste methode voor een voorspelalgoritme met seizoenseffect is redelijk simpel en vereist ook niet veel rekentijd. Toch zijn er een aantal problemen waardoor dit ‘seasonal decomposition’ algoritme niet als zodanig in ORION kan worden ingevoerd. In de paragrafen 5.1.1 tot en met 5.1.4 beschrijf ik deze problemen, met de oplossingen die ik hiervoor gevonden heb.

5.1.1 Toewijzen van het verbruik over maanden

Ik heb ervoor gekozen om op basis van maanden het seizoensprofiel op te zetten. Dit heeft de volgende redenen:

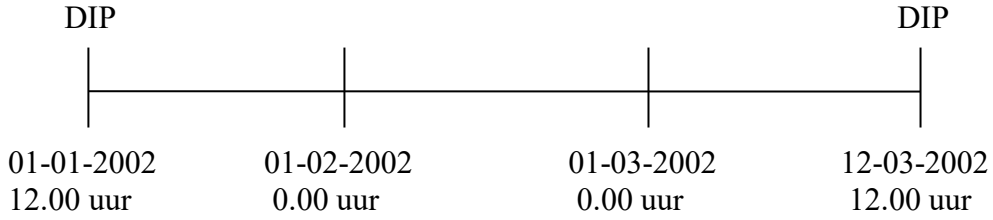
- Een kwartaalprofiel is niet nauwkeurig genoeg. Binnen een kwartaal zijn flinke schommelingen in het gasverbruik niet onwaarschijnlijk. Bijvoorbeeld door een winterpiek.
- Een profiel op weekbasis is te gedetailleerd. Het profiel zal niet van week tot week verschillen. Bovendien kan een profiel op weekbasis schijnnauwkeurigheid opleveren, omdat er in de gasmarkt veel minder DIPs zijn. Als er 3 maanden geen DIP is, is het niet duidelijk hoe groot het verbruik per week is.

De keuze voor maanden betekent dat de benodigde input voor het ‘seasonal decomposition’ algoritme, het absolute verbruik per maand is. Deze input is niet direct beschikbaar. De data die wel beschikbaar is zijn DIPs en DROPs. (In de gasmarkt zijn er geen SALES). In de huidige situatie verdeelt ORION het verbruik tussen DIPs over de verschillende dagen van de week. Dit levert een weekprofiel op. Op basis van dit weekprofiel is het voorspellen van een seizoenseffect per maand niet mogelijk. Daarom is een module nodig die het verbruik tussen DIPs toewijst aan de verschillende maanden van het jaar. Het verbruik naar maanden toewijzen gaat op een vrijwel identieke manier als de eerder beschreven toewijzingsmethode die nu in ORION zit.

- Als twee opeenvolgende DIPs in dezelfde maand plaatsvinden wordt het verbruik tussen deze DIPs toegevoegd aan het maandverbruik van deze maand.
- Anders: Het verbruik tussen twee DIPs wordt verdeeld over de tussenliggende maanden.

Ik zal deze toewijzingsmethode met een voorbeeld toelichten.

Voorbeeld:



Het totaal aantal dagen tussen de twee DIPs is: 70.

Van de eerste DIP tot eind januari is 30.5 dagen. Februari telt geheel mee (28 dagen), en van begin maart tot de tweede DIP is 11.5 dagen.

Het resultaat van de toewijzing is dan:

Januari: 30.5 / 70 * absoluut verbruik tussen de DIPs.
Februari: 28 / 70 * absoluut verbruik tussen de DIPs.
Maart: 11.5 / 70 * absoluut verbruik tussen de DIPs.

Op deze manier loop je de historie door van de eerste tot de laatste DIP. Nu heb je voor elke maand in de historie een verbruikscijfer, de benodigde input voor het ‘seasonal decomposition’ algoritme.

5.1.2 Buiten beschouwing laten van data

De eerste en laatste maand in de historie zijn niet compleet, tenzij de eerste en laatste DIP precies op een maandovergang plaatsvinden. Dit heeft tot gevolg dat de maandverbruiken die aan deze 2 maanden worden toegewezen ook niet volledig zullen zijn. Om te voorkomen dat het seizoenspatroon verkeerd wordt ingeschat heb ik er daarom voor gekozen de eerste en laatste maand buiten beschouwing te laten. Een andere optie is het opschalen van de verbruiksgegevens in deze maanden. Het nadeel van deze methode is dat de schalingsfactor heel groot kan worden. Als de laatste DIP in de historie op 1 augustus om 1:00 uur is, dan moet het verbruik voor augustus met een factor van 743 worden opgeschaald. (23 uur voor 1 augustus + 30 * 24 uur). Als er een onnauwkeurigheid in de data zit, dan wordt deze gigantisch opgeblazen. Vandaar de keuze om de eerste en laatste maand van de historie buiten beschouwing te laten.

5.1.3 Databehoefte en beschikbaarheid

Het 'seasonal decomposition' algoritme zoals beschreven in paragraaf 4.1 heeft tenminste 24 maanden verbruiksgegevens nodig om een seizoensprofiel te kunnen berekenen. Omdat de eerste en laatste maand buiten beschouwing worden gelaten komen hier nog eens 2 maanden bij. Nog meer data is te prefereren aangezien de seizoensratio's dan kunnen worden gemiddeld. In praktijk is deze beschikbaarheid van data soms een probleem. In Tabel 5.1 wordt aangegeven hoe het algoritme wordt toegepast bij een gegeven databeschikbaarheid.

Data beschikbaarheid	*Toepassing algoritme (na 'weggooien' eerste en laatste maand van de data)*
< 14 mnd.	Er is minder dan een jaar data beschikbaar. Te kort om het 'seasonal decomposition' algoritme toe te passen, of om iets zinnigs te kunnen zeggen over seizoenseffecten.
> 14 maar < 26 mnd.	Er is tussen de 1 en 2 jaar data beschikbaar. Te kort om het 'seasonal decomposition' algoritme toe te passen. Om toch te kunnen rekenen neem je voor de verwachting van het toekomstige verbruik, de echte verbruiksgegevens van vorig jaar. Als er van bepaalde maanden 2 verbruiksgegevens zijn, worden deze gemiddeld.
> 26 mnd.	Er is voldoende data beschikbaar. Het 'seasonal decomposition' algoritme kan worden toegepast.

Tabel 5.1: Gebruik van 'seasonal decomposition' bij verschillende data beschikbaarheid

5.1.4 Trend

Ik ga ervan uit dat er geen trend aanwezig is in de vraag naar benzine en/of gasproducten. Dit heeft tot gevolg dat na berekening van de seizoensindices het toekomstige verbruik door middel van 'exponential smoothing' berekend wordt. De redenen om aan te nemen dat er geen trend is zijn:

- Uit eerder onderzoek van J. Haasakker[1] blijkt dat het meenemen van een trend voor de benzinemarkt meestal niet zinvol is: 'Voor de meeste tanks geeft het voorspellen met trend geen betere voorspelling dan het voorspellen zonder trend'.
- In de door mij bekeken databases heb ik steekproefsgewijs verbruiksgrafieken van verschillende klanten bekeken. Hierin was geen trend waarneembaar.
- In de huidige versie van ORION wordt in de algoritmiek geen rekening gehouden met een trend. Om een goede vergelijking tussen de verschillende algoritmes mogelijk te maken, houden we hier ook in het nieuwe algoritme geen rekening mee.

[1] Zie J. Haasakker: Vraagvoorspelling en voorraadbeheersing voor benzinestations. [2] pag 31 - 37

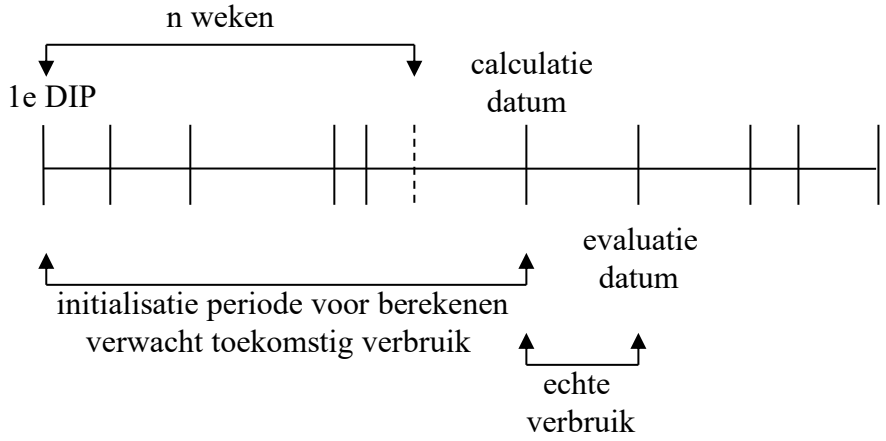
5.2 Oplosmethoden voor meten van de ‘performance’ van voorspelalgoritmes

Na het implementeren van een nieuw algoritme is het natuurlijk van belang te meten hoe goed dit algoritme ‘presteert’ in vergelijking met andere algoritmes. Hoe deze zogenaamde ‘performance’ moet worden gemeten is niet direct duidelijk. De uitvoer van ORION is een aantal gegenereerde orders, gebaseerd op de vraag in het verleden. Het is onmogelijk om aan te geven in hoeverre deze toekomstige orders ‘beter’ of ‘slechter’ zijn dan de orders gegenereerd door een ander algoritme. De toekomstige vraag naar de producten is natuurlijk nog niet bekend. Wel een mogelijkheid is om te vergelijken hoe de verschillende algoritmes in het verleden zouden hebben gepresteerd. De manier waarop ik dit heb aangepakt beschrijf ik in de volgende paragraaf.

5.2.1 Berekenen van afwijkingspercentages

Als maat voor de ‘performance’ van een algoritme neem ik de afwijkingen tussen werkelijke DIPs en de DIP waarden die ORION voorspelt op basis van de verwachte vraag. Ik zal aan de hand van een stappenplan de performance berekening uitleggen en uitleggen welke keuzes ik heb gemaakt.

Onderstaand staat een schema van DIPs. In dit schema staat n voor het aantal weken historie wat wordt meegenomen. Het schema is een visualisatie van de eerste stappen van het stappenplan.



Stappenplan voor het berekenen van afwijkingspercentages per klant:

- 1: Neem de eerste DIP uit de historie. Ga n weken vooruit in de tijd, en pak de eerstvolgende DIP hierna. Dit is de calculatie datum.

- 2: Bereken het verwachte toekomstige verbruik op basis van alle DIPs tussen de eerste DIP en de calculatie datum.

- 3: Ga naar de eerstvolgende DIP na de calculatiedatum. Noem dit de evaluatie datum. Bereken het werkelijke (absolute) verbruik tussen de calculatie en evaluatie datum.

- 4: Ga op basis van de DIP waarde op de calculatiedatum en het verwachte toekomstige verbruik na welke DIP waarde ORION verwacht op de evaluatie datum. Bereken het verwachte verbruik tussen calculatie en evaluatie datum.

- 5: Bereken het absolute verschil tussen het werkelijke en het verwachte verbruik. Noem dit de absolute afwijking.

Nadat dit stappenplan één keer is doorlopen verandert stap 1. De nieuwe stap 1 wordt:

- 1: De calculatie datum wordt gelijk aan de evaluatie datum. De nieuwe evaluatie datum wordt gelijk aan de datum van de eerstvolgende DIP na de vorige evaluatie datum. De eerste DIP die wordt gebruikt is nu de eerste DIP na (de calculatie datum – n weken).

De rest van de stappen kan gewoon worden doorlopen.

Door op deze manier het stappenplan te doorlopen, bereken je een rijtje met absolute afwijkingen tussen het werkelijke verbruik en het door ORION voorspelde verbruik. Deze afwijkingen zeggen nog niet zoveel. Er moeten nog een aantal bewerkingen op worden uitgevoerd. Dat zal ik toelichten met een voorbeeld.

Voorbeeld (de getallen in dit voorbeeld zijn fictief):

Absolute afwijkingen: (100, 123, 4620, 157, 58, 98, 327)

Op het eerste gezicht is de afwijking van 4620 dramatisch en heeft het algoritme daar heel slecht gepresteerd. Maar in de berekening van deze afwijkingen is geen rekening gehouden met de tijden tussen de DIPs. Deze tijden kunnen een heel grillig verloop hebben, het is dus niet ondenkbaar dat de afwijking van 4620 liter relatief gezien de kleinste afwijking per dag is. Er moet dus een stap aan het stappenplan worden toegevoegd.

- 6: Bereken de relatieve afwijkingen door de absolute afwijkingen te delen door het aantal dagen dat tussen de desbetreffende DIPs zit.

Nu hebben we een rijtje met relatieve afwijkingen. Deze waarden zeggen al wat meer, maar blijven moeilijk te interpreteren. Stel dat de relatieve afwijkingen er zo uit zien:

Relatieve afwijkingen: (140, 126, 311, 103, 212, 175, 98)

Op het eerste gezicht zitten er geen grote uitschieters meer tussen. Maar het is onmogelijk te beoordelen hoe goed het algoritme in dit geval heeft gepresteerd, als je het gemiddelde verbruik per dag niet weet. Als in het voorbeeld het gemiddelde verbruik 2000 liter per dag is, heeft het algoritme veel beter gepresteerd dan wanneer het gemiddelde verbruik 500 liter per dag is.

Stap 7 en 8 van het stappenplan worden:

- 7: Bereken de afwijkingspercentages per dag door de relatieve afwijkingen te delen door het gemiddelde verbruik per dag.

- 8: Bereken het gemiddelde afwijkingspercentage per tank, door de afwijkingspercentages per dag op te tellen en te delen door het aantal waarnemingen.

Na uitvoering van bovenstaand stappenplan is voor een klant voor al zijn tanks een gemiddeld afwijkingspercentage bekend. Bij de ene tank zal 'moving average' het misschien iets beter doen, terwijl bij een andere tank 'exponential smoothing' of 'seasonal decomposition' het iets beter doet. Om per klant te kunnen beoordelen welk algoritme beter presteert, moeten de gemiddelde afwijkingspercentages per tank worden teruggebracht naar één percentage per klant. De simpelste manier om dit te bereiken is het gemiddelde van de afwijkingspercentages per tank te nemen. Het nadeel hiervan is dat elke tank even zwaar meetelt in de berekening. Voor een tank die ongeveer dagelijks moet worden gevuld is een goede voorspelling veel belangrijker dan voor een tank waarvan de inhoud gemiddeld 2 weken meegaat. Daarom heb ik ervoor gekozen om een gewogen gemiddelde te nemen. Als wegingsfactor heb ik gekozen voor: (1 / het gemiddeld aantal dagen dat met een volle tank kan worden gedaan). Een tank waarvan de inhoud gemiddeld 1 dag meegaat telt dan 7x zo zwaar mee als een tank die gemiddeld eens per week gevuld moet worden. De laatste stap van de berekening is:

- 9: Bereken een gewogen gemiddelde afwijkingspercentage per algoritme. De

$$\text{wegingsfactor is: } \frac{\textit{gemiddelde verbruik per dag}}{\textit{maximum voorraad - veiligheidsvoorraad}}$$

Bovenstaand stappenplan heb ik geïmplementeerd in ORION. Door bovenstaand stappenplan voor de verschillende algoritmes uit te voeren krijg je per klant per algoritme een afwijkingspercentage. Door deze afwijkingspercentages te vergelijken kun je per klant zien welk algoritme in het verleden het best gepresteerd zou hebben. De aanname is dat dit algoritme het in de toekomst ook het beste doet voor die bepaalde klant.

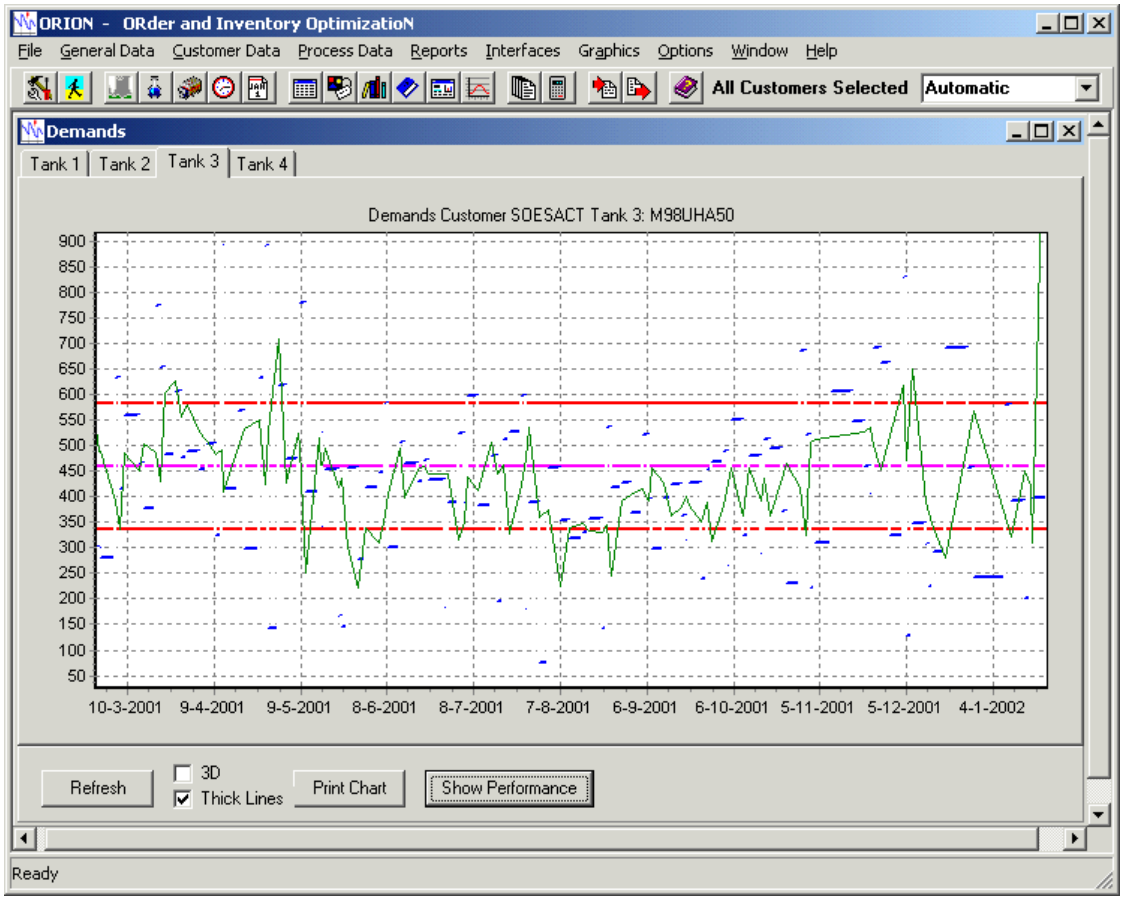
Het is ook mogelijk om in plaats van absolute verschillen de kleinste kwadraten methode te gebruiken. Dit is een veel gebruikte methode waarbij de afwijkingen worden gekwadrateerd om grotere afwijkingen zwaarder mee te laten tellen. Ik heb er voor gekozen dit niet te doen omdat er regelmatig uitschieters in de data voorkomen. Als zulke afwijkingen dan nog worden gekwadrateerd bepaalt één enkele waarde voor een groot deel de beoordeling van de voorspelling. Een voorbeeld van deze uitschieters staat in Bijlage C.

NB. In de berekening van de afwijkingspercentages wordt ook rekening gehouden met de 25-procentsregel. Omdat DIPs die hier niet aan voldoen niet worden meegenomen in de vraagvoorspelling, worden ze ook niet meegenomen in het berekenen van de afwijkingspercentages.

5.2.2 Visualisatie

In plaats van alleen afwijkingspercentages te berekenen heb ik de vraagvoorspelling ook gevisualiseerd. Er is in ORION al de mogelijkheid om een grafiek te tekenen van het werkelijke verbruik tussen DIPs. Dit verbruik is geschaald naar een gemiddeld verbruik per dag. Ik heb de code voor het maken van deze grafieken uitgebreid, zodanig dat de door ORION voorspelde vraag ook zichtbaar is in deze grafieken. Het levert interessante plaatjes op voor de gebruiker om de prestaties van de algoritmes te kunnen volgen. Zo kan de invloed van uitbijters op de vraagvoorspelling voor de verschillende algoritmes worden vergeleken. Deze grafieken bleken ook heel handig bij het debuggen van de code, omdat vreemde voorspellingen sneller opvallen.

In Figuur 5.1 staat een voorbeeld van zo'n grafiek.
De blauwe streepjes stellen de werkelijke vraag tussen 2 DIPs voor. De groene lijn is de voorspelde vraag. In dit voorbeeld is de vraag voorspeld met het 'exponential smoothing' algoritme.



Figuur 5.1: Voorbeeld van voorspelde vraag in grafiek

In bovenstaande figuur vallen twee dingen op:

- De vraag schommelt behoorlijk.
- Het algoritme zit er af en toe flink naast.

Het feit dat de voorspelling van het algoritme af en toe flink afwijkt van de werkelijke vraag kan worden verklaard door deze vraagschommelingen. Zie hiervoor bijlage D.

5.3 Oplosmethoden voor classificering van de benzinemarkt

Voor de benzinemarkt zijn alleen ‘exponential smoothing’ en ‘moving average’ relevant. Het ‘seasonal decomposition’ algoritme werkt op maandbasis en dat is voor de benzinemarkt te ‘grof’. Het doel van classificering van klanten voor de benzinemarkt is dus aangeven welke klanten hun vraagvoorspelling het beste met ‘exponential smoothing’ kunnen doen en welke met ‘moving average’.

5.3.1 Minimum databehoefte

In de benzinedatabase die ik analyseer staan zo’n 450 klanten. Ik heb ervoor gekozen om klanten met een historie van minder dan een jaar buiten beschouwing te laten. Dit om te voorkomen dat de performance wordt bepaald aan de hand van een handjevol DIPs.

5.3.2 Analyse vooraf

Om te beoordelen of classificatie überhaupt zinvol is, maak ik gebruik van de performance module die ik in paragraaf 5.2 heb beschreven. Voor een database met ruim 400 klanten bereken ik de performance van het ‘exponential smoothing’ en het ‘moving average’ algoritme. Het is extra interessant om deze 2 methodes eens te vergelijken, omdat dit voor ORION nog niet eerder gedaan is. Als de verschillen tussen de algoritmes heel klein zijn, of één algoritme altijd beter presteert dan het ander, is classificatie niet zinvol.

5.3.2.a Parameter instellingen

Voordat ik daadwerkelijk een performance run heb gedaan voor de benzinemarkt heb ik eerst op een testset wat resultaten bekeken. Daarbij maakte ik gebruik van de instellingen zoals ze al in de database stonden. In Bijlage E staat een samenvatting van de berekeningen op deze testset. Wat opvalt is:

- In de meeste gevallen geeft 'moving average' het beste resultaat. Zie ook bijlage E.

Omdat 'moving average' vaak veel beter presteerde op deze testset dan 'exponential smoothing' twijfelde ik aan de juistheid van de instelling van de parameter α in het 'exponential smoothing' algoritme. De oorspronkelijke instelling staat in Tabel 5.2.

α	*# dagen tussen de DIPs*
0.5	0 t/m 5
0.6	6 of 7
0.7	8 t/m 14
0.8	15 t/m 21
0.9	> 21

Tabel 5.2: oude α instelling

Als het aantal dagen tussen de DIPs kleiner dan 5 is, wordt α = 0.5 genomen. Is het aantal dagen tussen de DIPs 6 of 7, dan wordt α = 0.6 genomen enz.

Omdat in de benzinemarkt vaak regelmatig DIPs plaatsvinden, vaak minstens 2 keer per week, lijkt deze verdeling niet zo zinvol. Vrijwel altijd zal een α waarde van 0.5 genomen worden. Bovendien lijken deze waarden voor α relatief hoog om de volgende redenen:

- In de verbruiksgegevens treden veel schommelingen en regelmatig uitschieters op. De standaarddeviaties van het verbruik zijn relatief hoog. Bij een hoge α, is het algoritme gevoelig, dus worden al deze schommelingen en uitschieters gevolgd. Dit heeft tot gevolg dat het algoritme zich 'nerveus' gedraagt. Als α wat lager wordt genomen neigt het algoritme meer naar een gemiddelde. In bijlage F heb ik een screenshot opgenomen van verbruiksgegevens van een tank, waarin deze schommelingen te zien zijn.
- In praktijk blijkt de beste waarde voor α vaak tussen de 0.1 en 0.3 te liggen.

Daarom heb ik een aantal andere instellingen voor α uitgeprobeerd. Ik heb uit de BP-database die ik uiteindelijk ga analyseren willekeurig 19 klanten gehaald, om een schatting te maken voor welke α het 'exponential smoothing' algoritme goed werkt.

De parameterinstellingen heb ik als volgt gekozen:

INSTELLING 1:		*INSTELLING 2:*	
α	***# dagen tussen de DIPs***	***α***	***# dagen tussen de DIPs***
0.1	0 tot 0.5	**0.1**	0 tot 1
0.2	0.5 tot 1.0	**0.2**	1 tot 2
0.3	1.0 tot 1.5	**0.3**	2 tot 3
0.4	1.5 tot 2.0	**0.4**	3 tot 4
0.5	2.0 tot 2.5	**0.5**	4 tot 5
0.6	> 2.5	**0.6**	> 5

Tabel 5.3: Parameterinstellingen 'exponential smoothing'

Daarnaast heb ik nog 3 vaste instellingen uitgeprobeerd waarin het aantal dagen tussen de DIPs geen invloed heeft op de keuze van α.

Instelling 3: α = 0.1 ongeacht het aantal dagen tussen de DIPs.
Instelling 4: α = 0.2 ongeacht het aantal dagen tussen de DIPs.
Instelling 5: α = 0.3 ongeacht het aantal dagen tussen de DIPs.

Het bleek dat voor een vaste α van 0.2 de beste resultaten werden behaald. De volledige resultaten van deze parameterinstellingen staan in bijlage G.

Op basis van deze kleine testset en gezien het feit dat voor verschillende klanten de optimale α kan verschillen, is het natuurlijk niet zo dat α = 0.2 dé optimale waarde is.

Aanname: De performance van een klant in de benzinemarkt, berekend met het 'exponential smoothing' algoritme met α = 0.2, is een benadering van de performance van die klant berekend met een optimale α.

NB. Voor de gasmarkt is de oorspronkelijke instelling van α wel acceptabel. Er zitten vaak veel dagen tussen de DIPs en het algoritme moet snel reageren om in te kunnen spelen op de winterpiek. Daarom maak ik voor de benzinemarkt gebruik van α = 0.2 en voor de gasmarkt van de oorspronkelijke instelling zoals in Tabel 5.2.

5.3.3 Gebruik van de performance module voor classificatie

Eigenlijk doet de performance module niks anders dan een classificatie. Immers, voor een klant worden de afwijkingspercentages per algoritme bepaald. Deze percentages worden opgeslagen in de database. Ik heb daarna een module gemaakt die simpelweg kijkt naar de huidige instelling van de voorspelmethodiek. Vervolgens wordt deze instelling vergeleken met de afwijkingspercentages in de database. Als dit niet overeenkomt (dus er wordt bijvoorbeeld gerekend met 'moving average', maar het afwijkingspercentage van 'exponential smoothing' is kleiner), dan wordt de instelling van de voorspelmetodiek aangepast. Met één druk op de knop wordt zo de hele klantendatabase doorlopen en geclassificeerd.

In de classificatie heb ik een drempelwaarde opgenomen. Deze drempelwaarde kan naar eigen inzicht worden ingesteld, en is standaard 5%. De instelling van een voorspelmethodiek wordt alleen gewijzigd als de verwachte verbetering tenminste gelijk is aan de drempelwaarde.

5.4 Oplosmethoden voor classificering van de gasmarkt

5.4.1 Minimum databehoefte

Uit Tabel 5.1 valt af te lezen dat de minimumdatabehoefte om het 'seasonal decomposition' algoritme toe te passen 2 jaar is. Ik heb alleen klanten meegenomen met minimaal 5 jaar historie. 3 jaar voor initialisatie (zodat de seizoensindices gemiddeld worden, zie ook paragraaf 4.1.1) en tenminste 2 jaar om te testen. Dit om te voorkomen dat de performance berekend wordt op basis van enkele DIPs.

5.4.2 Opzetten van een seizoensprofiel met de graaddagen methode

Het 'moving average' algoritme in ORION kan worden uitgebreid met een handmatig opgezet seizoensprofiel. In dat geval stelt een gebruiker een aantal periodes in, met het percentage verbruik behorend bij die periode. Ik heb geprobeerd om een seizoensprofiel op te zetten dat representatief is voor de werkelijkheid. Daarom heb ik gebruik gemaakt van de temperatuursinformatie uit Tabel 5.4.

maand=>	jan	feb	mrt	apr	mei	jun	jul	aug	sep	okt	nov	dec
gem. min. temp. °C	2	1	3	4	7	9	11	11	9	7	4	3
gem. max. temp. °C	6	7	9	11	14	16	17	17	15	12	9	7

Tabel 5.4: Gemiddelde minimum en maximum temperaturen in Perth (Schotland). Bron: [7]

Dit zijn de klimatologische gemiddelden van minimum en maximum temperaturen in Perth (Schotland). De gasdatabase die ik onderzoek bevat Schotse klanten. Om van de minimum en de maximum temperaturen 1 getal te maken heb ik het gemiddelde van deze waarden genomen en dit afgerond. Dat levert:

maand=>	jan	feb	mrt	apr	mei	jun	jul	aug	sep	okt	nov	dec
gem. temp. °C	4	4	6	8	11	13	14	14	12	10	7	5

Tabel 5.5: Gemiddelde maandtemperatuur in Perth (Schotland)

Dit levert de volgende graaddagen op:

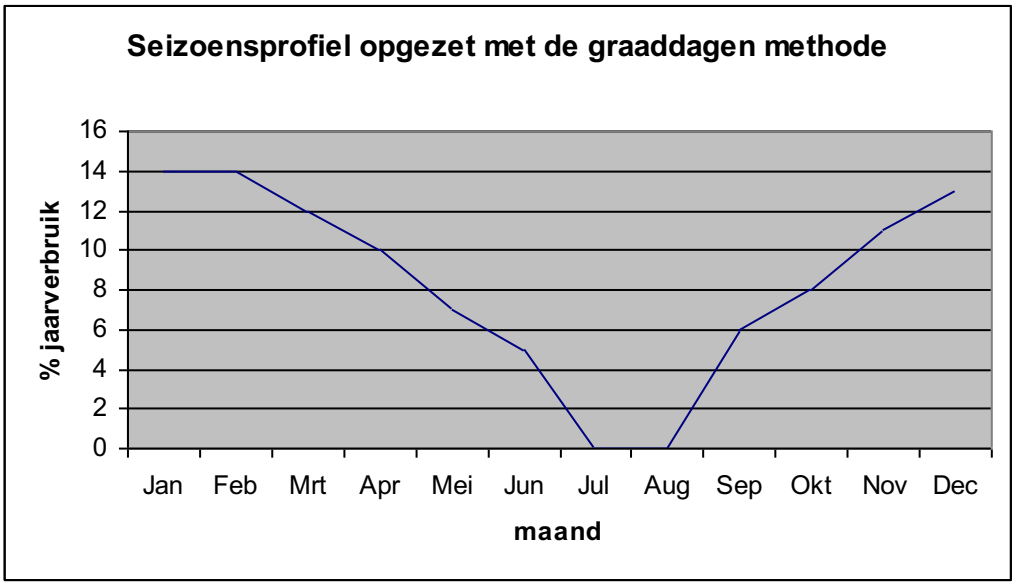
maand=>	jan	feb	mrt	apr	mei	jun	jul	aug	sep	okt	nov	dec
graaddagen	14	14	12	10	7	5	0	0	6	8	11	13

Tabel 5.6: Gemiddeld aantal graaddagen per maand in Perth (Schotland)

Het totaal aantal graaddagen is de som van de 12 maanden = 100.

De invoer in ORION per periode moet een percentage zijn van het totale jaarverbruik. Omdat de graaddagen toevallig exact sommeren tot 100, heb ik gewoon de graaddagen ingevoerd. Mochten de waarden niet sommeren tot 100 dan kunnen ze toch als zodanig worden ingevoerd, ORION schaalt de waarden automatisch, zodat het totaal op precies 100% komt.

Grafisch ziet dit seizoensprofiel er als volgt uit:



Figuur 5.2: Seizoensprofiel opgezet met de graaddagen methode

Dit seizoensprofiel zou een redelijk patroon moeten zijn voor de klanten in de gasdatabase die ik heb onderzocht, omdat deze database gegevens van Schotse klanten bevat.

6 Testresultaten

6.1 Vergelijking van ‘moving average’ en ‘exponential smoothing’

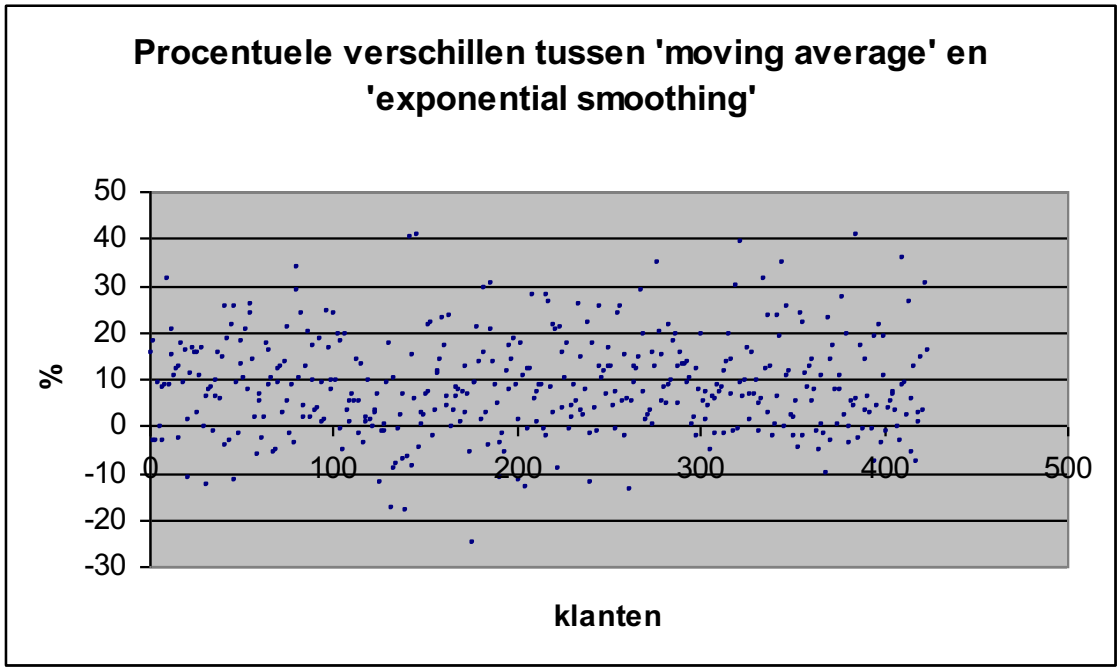
6.1.1 Samenvatting van de resultaten

De database die ik analyseer bevat 424 klanten. In Tabel 6.1 staat een samenvatting van de prestaties van de algoritmes.

Totalen	*Moving Average*	*Exponential Smooting*
424 klanten	333 x beste keus (78.5%) gemiddeld 11.8 % beter	91 x beste keus (21.5%) gemiddeld 4.6 % beter

Tabel 6.1: Prestaties van ‘moving average’ en ‘exponential smoothing’ voor de benzinemarkt

In ruim ¾ van de gevallen geeft het ‘moving average’ algoritme de beste resultaten. Als het ‘exponential smoothing’ algoritme al beter werkt, dan is het verschil relatief klein, gemiddeld 4.6%. Interessant is natuurlijk de spreiding rondom deze waarden. Die valt af te lezen in Figuur 6.1.



Figuur 6.1: Scatterplot van procentueel verschil tussen ‘moving average’ en ‘exponential smoothing’

Figuur 6.1 is een scatterplot van het procentuele verschil tussen de twee algoritmes per klant. De punten met een positieve waarde zijn waarnemingen waar ‘moving average’ beter werkt. De punten met een negatieve waarden zijn waarnemingen waar ‘exponential smoothing’ beter werkt.

Wat opvalt is:

- Als ‘exponential smoothing’ beter werkt, is het percentage verbetering kleiner dan als ‘moving average’ beter werkt.

Eerst zal ik kijken hoe grote verschillen in het voordeel van ‘moving average’ te verklaren zijn. Vervolgens zal ik kijken hoe grote verschillen in het voordeel van ‘exponential smoothing’ te verklaren zijn. Tenslotte kijk ik waarom de verbetering kleiner is als ‘exponential smoothing’ beter werkt dan ‘moving average’.

NB. In alle grafieken in deze paragraaf heb ik ingezoomd om de resultaten beter te kunnen laten zien. Dit is de verklaring voor het feit dat er af en toe vreemd uitziende uitschieters in de voorspelling te zien zijn. Op deze punten reageren de algoritmes op een verbruikscijfer wat in de ingezoomde grafiek niet zichtbaar is.

6.1.2 Grote verschillen in het voordeel van ‘moving average’

De relatief grote verschillen in het voordeel van ‘moving average’ zijn te verklaren door het weekprofiel. Dat zal ik hieronder toelichten.

Voorbeeld:

Bij klant ‘HERTBRACPM’ ziet het weekprofiel voor tank 2 er als volgt uit:

Gem.	Zondag	Maandag	Dinsdag	Woensdg	Donderdg	Vrijdag	Zaterdag
10314	7037	8155	12456	13466	12950	11205	7440

In het weekend is duidelijk veel minder verbruik dan midden in de week. Er is een duidelijk weekprofiel zichtbaar.

Onderstaande afwijkingspercentages zijn de afwijkingen berekend met de performance module. ‘Verschil’ is het procentuele verschil tussen de twee afwijkingspercentages.

Afwijkingspercentages:	‘moving average’:	27.45%
	‘exponential smoothing’:	34.47%
	verschil:	20.40%

Het ‘exponential smoothing’ algoritme kijkt niet naar dagen van de week. Stel nu dat de laatste twee DIPs hebben plaatsgevonden op dinsdag en donderdag. Tussen die dagen was het verbruik hoog, dus het exponential smoothing algoritme voorspelt voor de komende periode ook een hoog verbruik. Dat terwijl in het weekend het verbruik duidelijk lager ligt. Het ‘exponential smoothing’ algoritme maakt bij het voorspellen van de vraag voor deze tank dus relatief grote fouten.

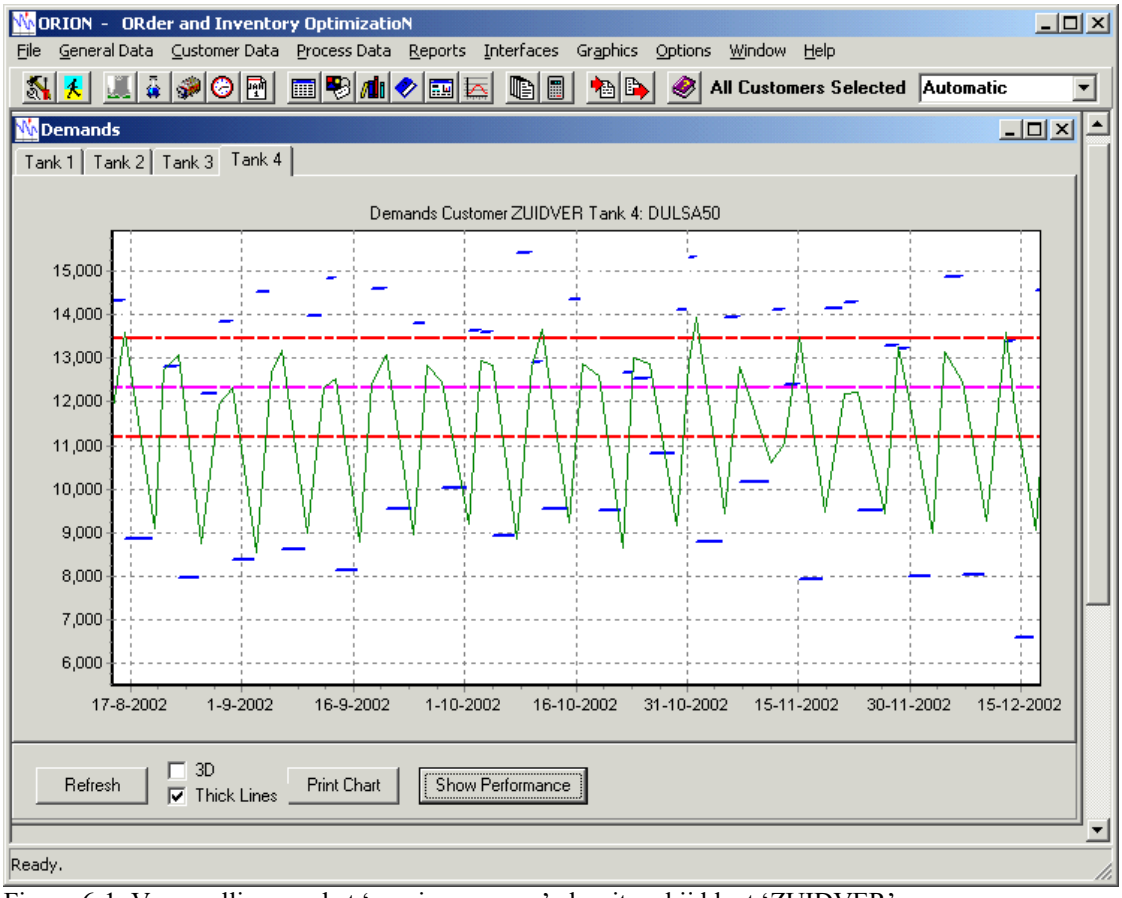
Daarentegen wordt bij het ‘moving average’ algoritme in ORION wel rekening gehouden met deze verschillen in de vraag per weekdag. Zie ook paragraaf 2.1.2. Daarom maakt het ‘moving average’ algoritme in dit geval relatief veel kleinere fouten.

Aangezien tank 2 in dit specifieke voorbeeld een gewicht van 75% heeft telt dit heel zwaar mee. Vandaar dat voor deze klant ‘moving average’ het veel beter doet dan ‘exponential smoothing’.

Dit zijn dus verklaarbare verschillen. Het zijn dus wel uitschieters, maar wel valide waarnemingen. Dit weekpatroon treedt eigenlijk altijd op bij dieseltanks. In de weekenden is de vraag naar diesel veel minder omdat er dan praktisch geen vrachtverkeer op de weg is.

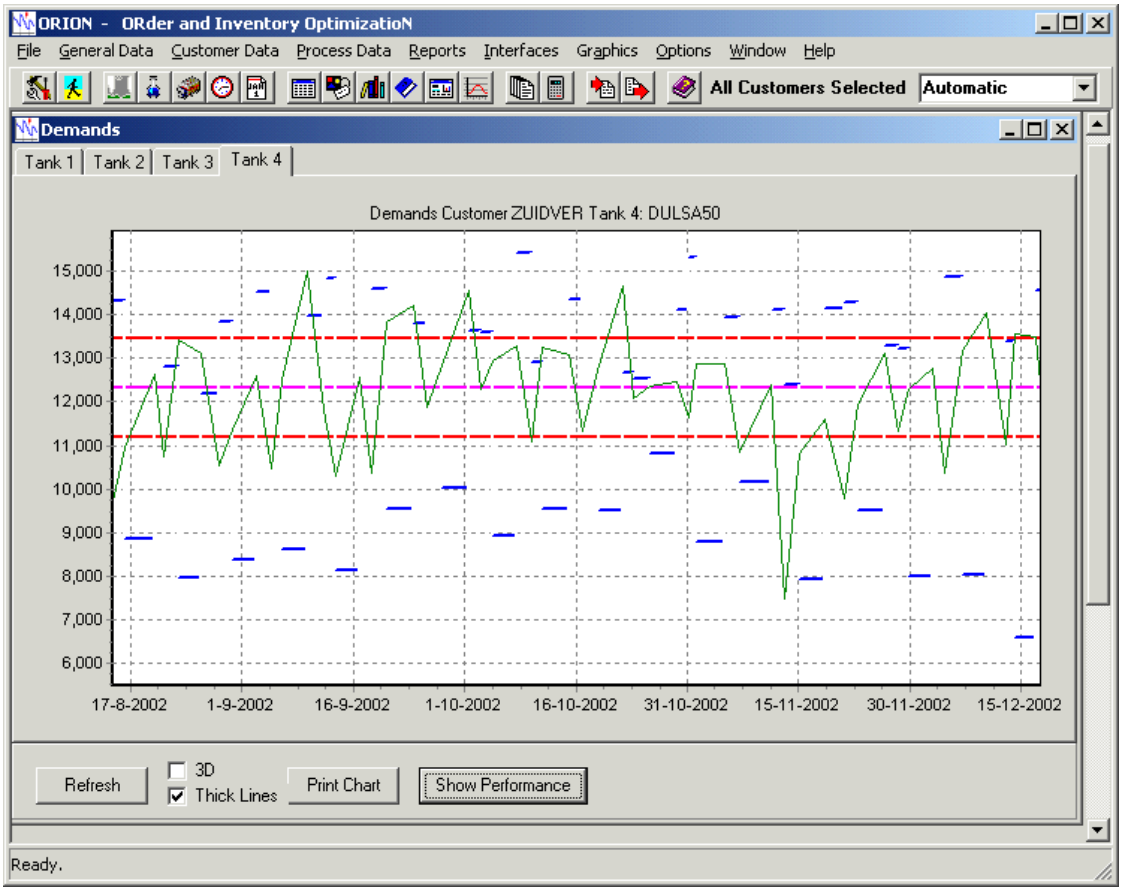
In een grafiek is het verschil in voorspellingen bij zo’n weekprofiel duidelijk te zien.

Figuur 6.1 is een grafiek van de vraag naar diesel bij klant ‘ZUIDVER’ en de voorspelling van het ‘moving average’ algoritme. Het weekprofiel is duidelijk te zien. De vraag verspringt telkens van hoog (doordeweekse dagen) naar laag (in het weekend). De voorspelling van het ‘moving average’ algoritme is behoorlijk goed, omdat er rekening wordt gehouden met het weekprofiel. De voorspelling gaat dus op de juiste momenten omhoog en naar beneden.



Figuur 6-1: Voorspelling van het ‘moving average’ algoritme bij klant ‘ZUIDVER’

Bij dezelfde tank zien de resultaten van het ‘exponential smoothing’ algoritme er veel minder goed uit. Zie hiervoor Figuur 6.2



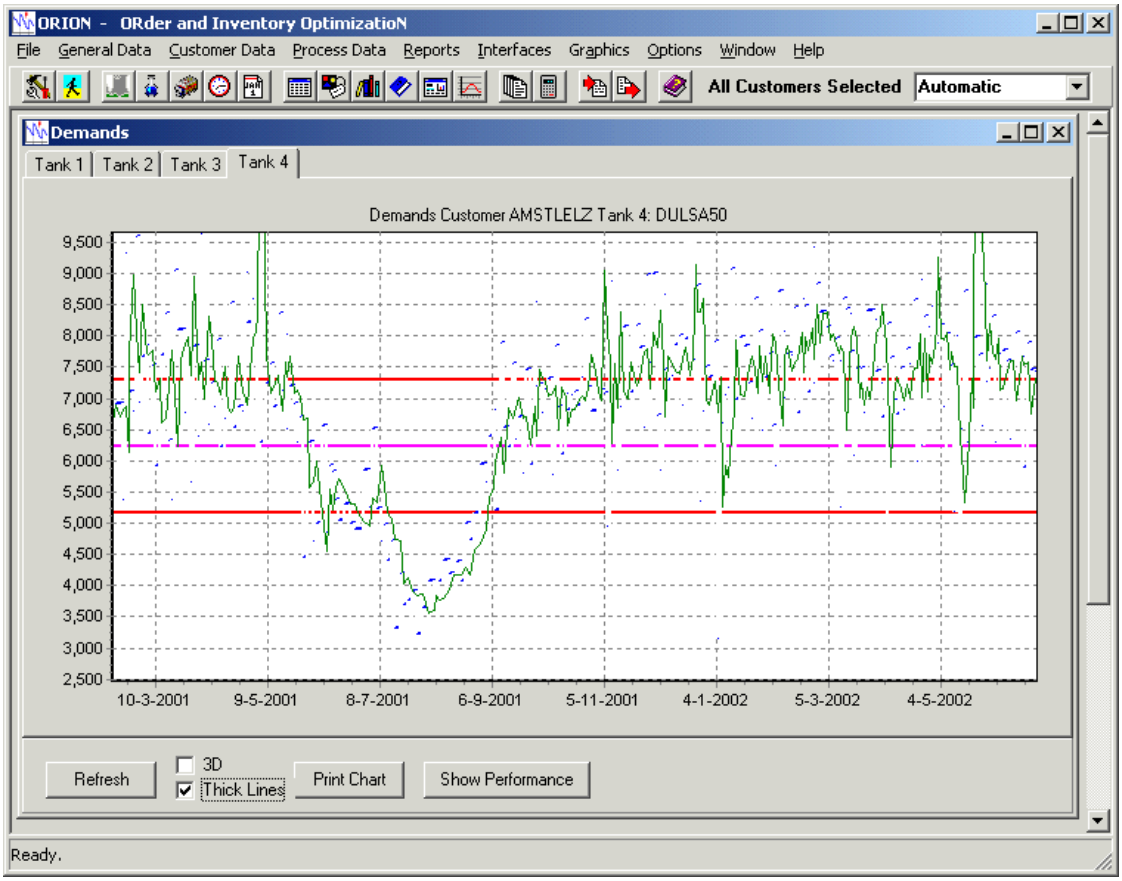
Figuur 6.2: Voorspelling van het ‘exponential smoothing’ algoritme bij klant ‘ZUIDVER’

Het ‘exponential smoothing’ algoritme reageert op de veranderingen in de vraag. Omdat er geen rekening wordt gehouden met een weekprofiel, maar de voorspelde vraag alleen afhankelijk is van de vraag in de vorige periode, is het algoritme iedere keer een stap te laat. De voorspelling is in dit geval dus een stuk slechter dan de voorspelling met het ‘moving average’ algoritme.

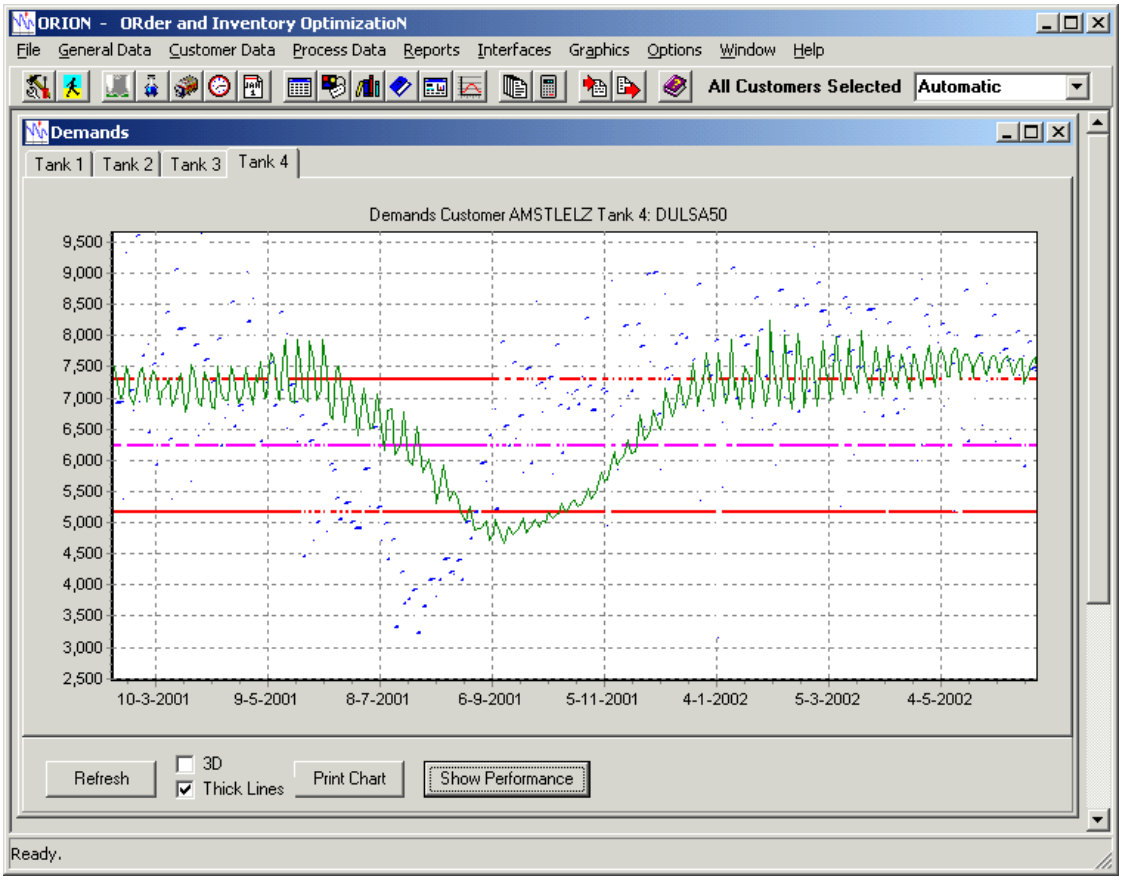
6.1.3 Grote verschillen in het voordeel van ‘exponential smoothing’

Ook de grote verschillen in het voordeel van ‘exponential smoothing’ zijn te verklaren. Vaak komt dit door een tijdelijke structurele verandering in de vraag. Dit is in een grafiek heel mooi te zien.

Figuur 6.3 laat de vraag zien naar diesel bij klant ‘AMSTLELZ’ en de bijbehorende voorspelling van het ‘exponential smoothing’ algoritme. Het is goed te zien dat vanaf ongeveer half mei tot half september 2001 de vraag structureel lager is. Een wegopbreking zou een oorzaak voor dit patroon kunnen zijn. Het ‘exponential smoothing’ algoritme reageert snel op deze verandering in de vraag en blijft dus redelijk goed voorspellen.



Figuur 6.3: Voorspelling van het ‘exponential smoothing’ algoritme bij klant ‘AMSTLELZ’



Figuur 6.4: Voorspelling van het ‘moving average’ algoritme bij klant ‘AMSTLELZ’

Figuur 6.4 laat een duidelijk ander beeld zien. De vraag is uiteraard exact gelijk, maar de voorspelling van het ‘moving average’ algoritme is een stuk slechter. Het duurt veel langer voordat het algoritme zich heeft aangepast aan de structurele verandering in de vraag. Voor deze specifieke klant is n in het ‘moving average’ algoritme gelijk aan 15 weken, dus het duurt behoorlijk lang voordat een verandering wordt meegenomen.

NB. Hieruit valt te concluderen dat het belangrijk is dat pomphouders de ‘special days’ functionaliteit zo goed mogelijk proberen te gebruiken. In paragraaf 2.1.1.f heb ik al aangegeven dat voor bepaalde dagen kan worden aangegeven dat de verwachte verkoop hoger of lager is dan normaal. Als een pomphouder zo’n structurele verlaging zoals in Figuur 6.3 ziet aankomen, bijvoorbeeld door een wegopbreking, dan is het van belang dat dit in ORION wordt ingesteld. Dan worden de voorspellingen een stuk beter.

6.1.4 Verschil in percentage verbetering

Het feit dat het percentage verbetering kleiner is als ‘exponential smoothing’ beter werkt is ook te verklaren door het weekprofiel. Bij diesel is er altijd sprake van een weekprofiel. Dit komt voornamelijk door vrachtverkeer, op doordeweekse dagen wordt daardoor veel meer diesel getankt dan in het weekend. Hierdoor werkt ‘moving average’ altijd beter voor de diesel tanks. Bij tanks waar dit weekprofiel niet speelt, werkt soms ‘moving average’ en soms ‘exponential smoothing’ iets beter. Als de ‘exponential smoothing’ methode het beste werkt, zal deze verbetering relatief niet zo groot zijn omdat voor de diesel tanks bij de klant ‘moving average’ beter werkt. De betere werking bij andere tanks moet dus groter zijn dan de slechtere werking bij de diesel tank, anders werkt het ‘exponential smoothing’ algoritme niet beter. Hoe groter het gewicht van de dieseltanks, hoe groter het voordeel voor het ‘moving average’ algoritme zal zijn. Hieruit blijkt meteen dat het eigenlijk helemaal niet zinvol is om de voorspelmethodiek per klant in te stellen. Het zou beter zijn dit per product of tank in te kunnen stellen.

6.2 Classificering van de benzinemarkt

Op basis van de resultaten uit paragraaf 6.1 valt te concluderen dat classificering voor de benzinemarkt misschien niet zo heel zinvol is. Voor een groot deel van de klanten werkt ‘moving average’ immers het best. Bovendien is de verbetering relatief klein als ‘exponential smoothing’ beter werkt. Toch is het, voor de kleine groep klanten waarbij ‘exponential smoothing’ beter werkt wel goed om te kunnen classificeren. Hoe klein de verbetering ook is, het is een verbetering. Op basis van de performance module worden de klanten geclassificeerd naar de te gebruiken voorspelmethodiek. Uit Tabel 6.1 valt af te lezen dat voor ¾ van de klanten de classificatie ‘moving average’ is, en voor ¼ van de klanten ‘exponential smoothing’.

6.3 De werking van ‘seasonal decomposition’ voor de gasmarkt

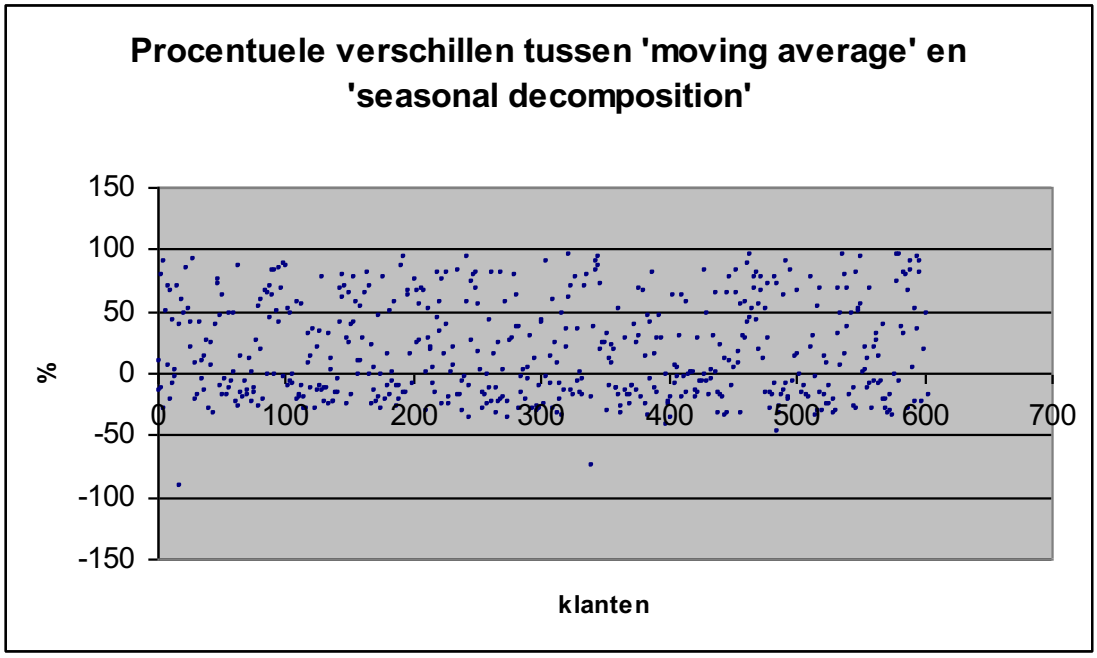
De gasdatabase die ik heb geanalyseerd bevat 604 klanten. In Tabel 6.2 staat een samenvatting van de prestaties van de algoritmes:

Totalen	*Moving Average*	*Seasonal Decomposition*	*Exponential Smooting*
604 klanten	245x beste keus (40.6%) gemiddeld 18.1 % beter	295x beste keus (48.8%) gemiddeld 46.8 % beter	64 x beste keus (10.6%) gemiddeld 6.3 % beter

Tabel 6.2: Prestaties van voorspelmethodieken voor de gasmarkt

In bijna de helft van de gevallen werkt het nieuw geïmplementeerde ‘seasonal decomposition’ algoritme beter dan beide andere algoritmes. Dit is natuurlijk een heel mooi resultaat. Wat opvalt is de slechte prestatie van het ‘exponential smoothing’ algoritme. De verklaring daarvoor is heel eenvoudig. Zowel het ‘seasonal decomposition’ algoritme als het ‘moving average’ algoritme houden rekening met een seizoensprofiel. Het ‘exponential smoothing’ algoritme doet dit niet, maar reageert op de veranderingen in de vraag. Daarom loopt de voorspelling met het ‘exponential smoothing’ algoritme iedere keer net één stap achter de feiten aan.

De spreiding van het verschil tussen ‘moving average’ en ‘seasonal decomposition’ staat in Figuur 6.5. De positieve waarden zijn waarnemingen waar ‘seasonal decomposition’ beter werkt.

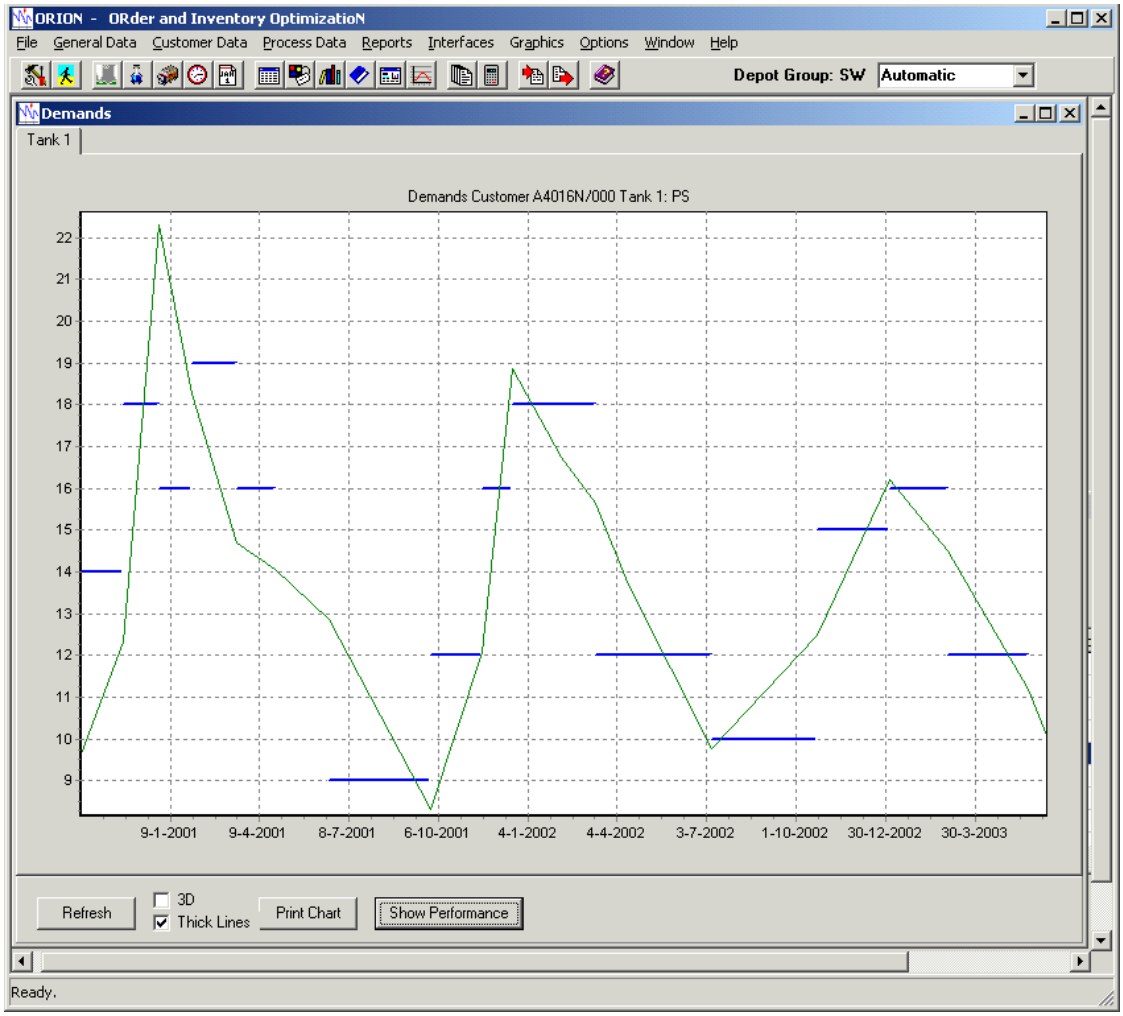


Figuur 6.5: Scatterplot van procentueel verschil tussen ‘moving average’ en ‘seasonal decomposition’

De verschillen tussen beide algoritmes zijn soms erg groot. Dit heeft een aantal oorzaken.

- Er wordt gerekend met heel weinig DIPs. Foute waarden in de database, zoals te hoge of te lage waarden voor het verbruik hebben daarom een hele grote invloed. Deze extreme waarnemingen komen regelmatig voor en vertroebelen de database, dus ook de resultaten van de performance module. De extreme waarnemingen worden maar over een beperkt aantal DIPs uitgespreid omdat er in de gasmarkt zo weinig DIPs zijn.
- Als er wel een seizoenspatroon in de data zit, maar volgens andere verhoudingen dan volgens het zelf opgezette seizoenspatroon van Figuur 5.3 dan zal 'moving average' een veel slechter resultaat geven dan 'seasonal decomposition'. Dit kan voorkomen als het seizoenspatroon ook van andere factoren dan alleen het weer afhankelijk is.

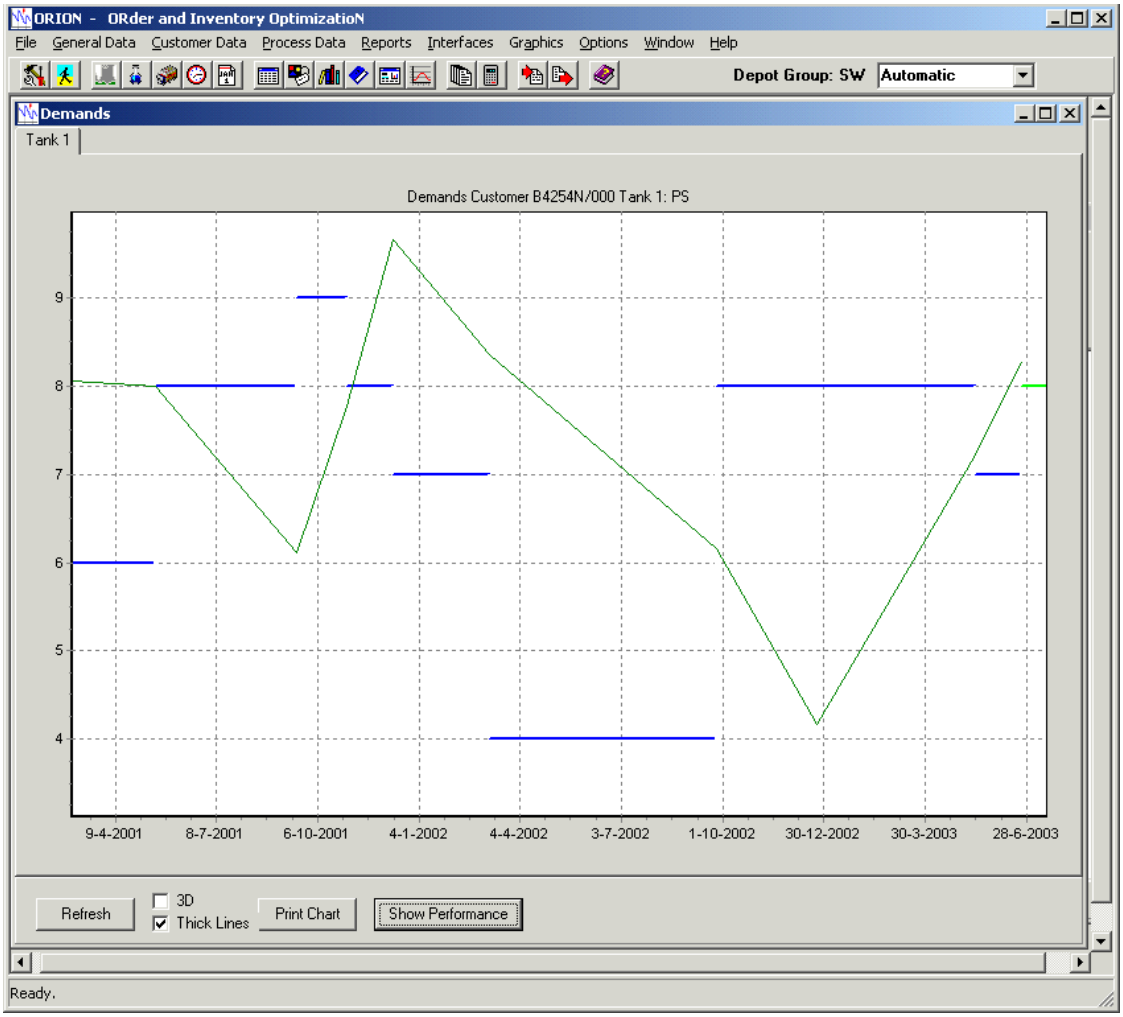
Het resultaat van een voorspelling van het 'seasonal decomposition' algoritme ziet er als volgt uit:



Figuur 6.6: Voorspelling met het 'seasonal decomposition' algoritme voor een gasklant (1)

In Figuur 6.6 is een mooi voorbeeld te zien van een voorspelling met het 'seasonal decomposition' algoritme. Het algoritme speelt goed in op de winterpieken bij de betreffende klant en heeft dus een goede performance.

Een verklaring voor het feit dat ‘seasonal decomposition’ in veel gevallen toch minder presteert dan ‘moving average’ heeft te maken met de plaats van de winterpieken. Als de winter vroeger of later invalt dan gemiddeld gedurende de drie jaren ervoor (initialisatieperiode is 3 jaar) dan zit het ‘seasonal decomposition’ algoritme er iedere keer net naast. Een voorbeeld hiervan is te zien in Figuur 6.7



Figuur 6.7: Voorspelling met het ‘seasonal decomposition’ algoritme voor een gasklant (2)

In deze figuur is te zien dat het ‘seasonal decomposition’ model achter de feiten aanloopt. Kennelijk viel de winterpiek bij deze gebruiker in dit jaar eerder dan de gemiddelde piek over de laatste 3 jaar. Dat is meteen het zwakke punt van de ‘seasonal decomposition’ methode. Er wordt met een soort van ‘standaardjaar’ gewerkt. Als blijkt dat de winter eerder of later invalt, dan worden de resultaten slecht. Hier zal ik in paragraaf 8.2 op terugkomen.

Een andere verklaring voor een betere werking van ‘moving average’ is het aantal DIPs. Bij maar een aantal DIPs per jaar, is bij het ‘seasonal decomposition’ algoritme het verbruik per maand onnauwkeurig. Voor alle maanden tussen 2 DIPs wordt hetzelfde verbruik verondersteld. Bij het ‘moving average’ algoritme wordt er nog rekening gehouden met het ingestelde seizoensprofiel, dus ook voor maanden waarin geen DIPs plaatsvinden worden de seizoenseffecten meegenomen.

7 Conclusies

7.1 Conclusies voor de voorspelalgoritmes

Een algemene conclusie die voor de voorspelalgoritmes getrokken kan worden is dat het ‘exponential smoothing’ algoritme maar in weinig gevallen het beste algoritme is om mee te voorspellen. Als pomphouders zorgvuldig omgaan met de ‘special days’ functionaliteit, zal het ‘moving average’ algoritme in nog meer gevallen beter werken dan het ‘exponential smoothing’ algoritme. Een belangrijke tweede conclusie voor de voorspelalgoritmes is dat het middelen van de afwijkingspercentages van alle tanks bij een klant eigenlijk onzin is. Het zou beter zijn om per tank of per product het beste voorspelalgoritme te in te kunnen stellen.

7.1.1 De benzinemarkt

Het is gebleken dat voor de benzinemarkt het ‘moving average’ algoritme over het algemeen het beste werkt. Bij de BP-database gaat dit in ¾ van de gevallen op. Als ‘exponential smoothing’ wel beter werkt is die verbetering relatief klein, gemiddeld zo’n 5%. Het nieuw geïmplementeerde ‘seasonal decomposition’ algoritme is niet geschikt voor de benzinemarkt. Ik heb dit algoritme daarom niet meegenomen in de tests voor de benzinemarkt.

7.1.2 De gasmarkt

Voor de gasmarkt valt te concluderen dat het moeilijk blijft om goede voorspellingen te doen. Toch heeft het ‘seasonal decomposition’ algoritme wat ik heb geïmplementeerd zeker toegevoegde waarde. In de helft van de gevallen is de voorspelling verbeterd door gebruikmaking van dit nieuwe algoritme. De verbeteringen zijn over het algemeen ook behoorlijk fors. Dit is een heel mooi resultaat. Desalniettemin is ook het ‘seasonal decomposition’ algoritme, zoals alle voorspelmethodieken, afhankelijk van de aangeleverde data. Voor klanten met erg weinig DIPs blijft het lastig voorspellen.

7.2 Conclusies voor classificering

De classificeringsmodule doet eigenlijk niks anders dan het automatisch verwerken van de resultaten van een performance run. Voor klanten in de database die op een ander algoritme staan ingesteld dan het algoritme dat volgens de performance module de beste keus is, wordt het algoritme automatisch aangepast. In principe werkt deze manier van classificeren goed. Een nadeel is dat de classificatie module zoals die nu is geïmplementeerd niet automatisch verschillende parametercombinaties kan testen. Dit zou een stap voorwaarts zijn. Hier kom ik in paragraaf 8.3 op terug.

8 Aanbevelingen

In dit hoofdstuk bespreek ik een aantal aanbevelingen en interessante punten voor een eventueel vervolgonderzoek. Dit zijn punten waar ik tijdens de uitvoering van mijn onderzoek tegenaan ben gelopen. Deels vallen ze buiten de 'scope' van mijn scriptie, of zijn ze niet uitvoerbaar binnen de duur van de stage.

8.1 Gebruik van 'special days' functionaliteit

Zoals eerder al opgemerkt is het voor de planners belangrijk dat de 'special days' voor de klanten zorgvuldig worden ingesteld. Anders kan het voorkomen dat bij een station waar wordt voorspeld met 'moving average', de kwaliteit van de voorspelling drastisch afneemt als er structurele veranderingen in de vraag zijn. Het duurt een tijd voordat het 'moving average' algoritme op zo'n vraagverandering reageert. Het is dus van belang dat pomphouders wanneer ze weten dat er in een bepaalde periode een structurele vraagverandering is, dit doorgeven aan de planners. Op deze manier kan de performance van het 'moving average' algoritme verbeteren. Pomphouders moeten het inzicht krijgen dat dit van belang is, anders zal er niet veel veranderen.

8.2 Een standaard jaar

Het besproken algoritme met seizoenseffecten berekent een seizoensprofiel door gebeurtenissen uit het verleden te middelen. De seizoensindices worden dus op basis van een 'gemiddeld jaar' berekend. Dit zal een goed uitgangspunt vormen voor de voorspelling in ORION. Maar eigenlijk is het niet reëel om dit gemiddelde jaar blindelings over te nemen. In de gasmarkt is de afname heel erg afhankelijk van het weer. Als de winter heel vroeg invalt, dan zal het patroon er dat jaar anders uit zien dan tijdens een 'gemiddeld jaar'. Behalve de beschreven methode om de seizoenseffecten te schatten is, lijkt het nuttig om in bepaalde gevallen de voorspelling te kunnen beïnvloeden. Het hoeft natuurlijk niet zo ver te gaan om het weer te voorspellen, maar als gedurende het jaar blijkt dat de winter zacht of juist heel koud wordt, dan kan deze informatie gebruikt worden om de voorspelling aan te passen.

In welke gevallen de voorspelling aangepast kan/moet worden en hoe deze aangepast moet worden zou in een vervolgonderzoek bekeken kunnen worden.

8.3 Parameters meenemen bij classificatie

Zoals de classificatie nu geïmplementeerd is, moet eerst een performance run worden gedraaid, vervolgens kan geclassificeerd worden. Nadeel van deze aanpak is dat de instelling van parameters niet wordt meegenomen in de classificatie. Als een aanpassing in de parameters wordt gedaan, zal hier bij het classificeren geen rekening mee worden gehouden als er niet eerst een nieuwe performance run wordt gedaan. Het is gebruiksvriendelijker als behalve een afwijkingspercentage ook de bijbehorende parameter in de database wordt opgeslagen bij een performance run. In dat geval kun je bijhouden bij welke parameter instelling een algoritme het beste presteert. De classificatie module zou dan zodanig uitgebreid moeten worden dat bij de classificatie ook de parameters kunnen worden aangepast. Nadeel van deze aanpak is dat het eigenlijk niet goed past in de huidige structuur van ORION. In deze structuur kunnen de parameters namelijk niet klantspecifiek ingesteld worden.

8.4 Klantniveau versus tankniveau

In de huidige structuur van ORION wordt de voorspelmethodiek op klantniveau ingesteld. Het kan voorkomen dat bij een klant met meerdere tanks de beste voorspelmethodiek per tank verschilt. Uit de performance berekeningen die ik heb gedaan blijkt dit ook regelmatig voor te komen. Zeker door het al genoemde weekprofiel, waarbij voor diesel tanks ‘moving average’ altijd beter werkt, is het middelen van de afwijkingspercentages van verschillende tanks eigenlijk onzin. Door de voorspelmethodiek op tankniveau in te stellen, kunnen de voorspellingen nog verbeteren. Dit is dus aan te bevelen. Er zal dan wel een afweging gemaakt moeten worden of deze verbetering opweegt tegen de kosten om deze aanpassing in de structuur van ORION aan te brengen.

9 Literatuurlijst

[1] Dr. G. P. Zhang. College: '*Generalized Modeling Techniques with Applications. Part 2: Forecasting.'* (2000) Zie ook: http://www.gsu.edu/~dscgpz/chap2/chapter2.ppt

[2] J. Haasakker. Scriptie: *'Vraagvoorspelling en voorraadbeheersing voor benzinestations.'* (2001)

[3] ORTEC consultants ORION user guide version 2.2

[4] Dr. M.C.M de Gunst Collegedictaat *'Statistical Models'* (Najaar 2001)

[5] Nuon website http://www.mc-wetter.de/nuon/evcnuon-berekening.html

[6] Time Series Analysis and Forecasting http://myphliputil.pearsoncmg.com/student/bp_newbold statsbuse_5/ch17.ppt

[7] Klimaatcijfers Groot Brittanië http://www.meteonet.nl/klimaat/klmeu_uk.htm

10 Bijlagen

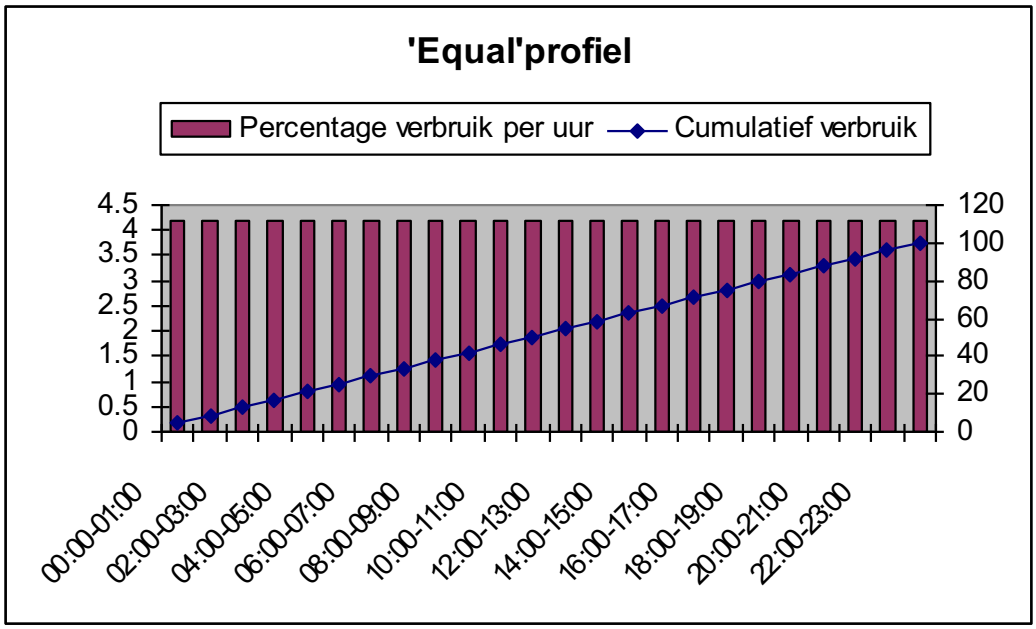
10.1 Bijlage A: Tabel en grafiek van ‘Equal’profiel

Bij het ‘Equal’profiel is de dagvraag evenredig verdeeld. Zie Tabel A1.

Tijd	*Percentage*	*Tijd*	*Percentage*
00:00-01:00	4.167	12:00-13:00	4.167
01:00-02:00	4.167	13:00-14:00	4.167
02:00-03:00	4.167	14:00-15:00	4.167
03:00-04:00	4.167	15:00-16:00	4.167
04:00-05:00	4.167	16:00-17:00	4.167
05:00-06:00	4.167	17:00-18:00	4.167
06:00-07:00	4.167	18:00-19:00	4.167
07:00-08:00	4.167	19:00-20:00	4.167
08:00-09:00	4.167	20:00-21:00	4.167
09:00-10:00	4.167	21:00-22:00	4.167
10:00-11:00	4.167	22:00-23:00	4.167
11:00-12:00	4.167	23:00-24:00	4.167

Tabel A1: Het ‘Equal’profiel

Figuur A1 geeft de grafiek van het ‘Equal’profiel



Figuur A1: Grafiek van ‘Equal’profiel

10.2 Bijlage B: Dataset voor vergelijken voorspelalgoritmes

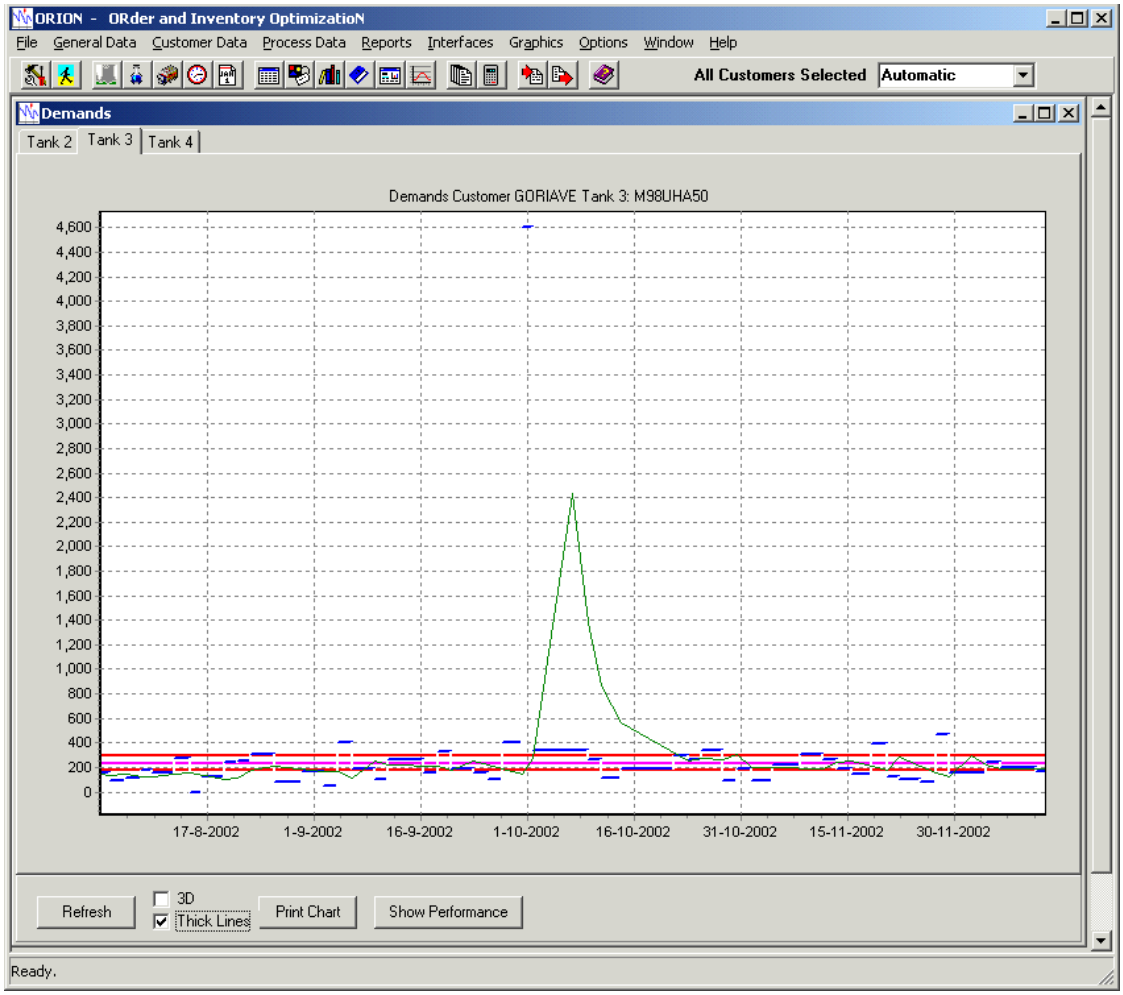
Voor het vergelijken van het moving average algoritme en het exponential smoothing algoritme heb ik het volgende datasetje gemaakt:

week	vraag
1	120
2	100
3	110
4	115
5	90
6	120
7	125
8	105
9	100
10	110
11	85
12	95

Dit levert de volgende voorspellingen op:

	MOVING AVERAGE				EXPONENTIAL SMOOTHING	
week	vraag	n=3	n=5		vraag	voorspelling met alfa = 0.5
1	120				120	
2	100				100	120
3	110				110	110
4	115	110			115	110
5	90	108.33			90	112.5
6	120	105	107		120	101.25
7	125	108.33	107		125	110.63
8	105	111.67	112		105	117.81
9	100	116.67	111		100	111.41
10	110	110	108		110	105.70
11	85	105	112		85	107.85
12	95	98.33	105		95	96.43

10.3 Bijlage C: Uitschieters



In bovenstaande figuur is het effect van een uitschieter op het ‘exponential smoothing’ algoritme te zien. De vraagvoorspelling schiet na de extreme waarneming van de vraag omhoog. Als ik de kleinste kwadraten methode zou toepassen in het berekenen van de performance, dan zou deze waarneming zoveel invloed hebben dat de andere afwijkingen niet relevant meer zijn. Dat is de reden dat ik werk met de absolute afwijkingen, zonder hier nog een wegingsfactor aan te hangen.

10.4 Bijlage D: Afwijkingspercentage vs. Standaarddeviatie

De afwijkingspercentages liggen voor de benzinemarkt ongeveer tussen de 10% en 30%. Dit lijkt behoorlijk hoog, maar dit is te verklaren door de hoge standaarddeviatie van de waarnemingen. Hoe hoger de standaarddeviatie, hoe groter de onvoorspelbaarheid, dus hoe hoger de afwijkingspercentages. Ik zal dit laten zien aan de hand van 2 voorbeelden uit de data.

Voorbeeld 1:

Tanknummer	0	1	2	3
Gem. afwijking per dag	13.51	436.13	99.29	185.13
Gem. verbruik per dag	36.08	3320.99	219.86	1230.86
Afwijkingspercentage	37.45	13.13	45.16	15.04
Wegingsfactor	7.17	42.14	26.22	24.47

Totaal Gewogen Afwijkingspercentage: 29.54

De standaarddeviaties per tank zijn voor deze klant:

Tank No	St. Dev (liters)
0	17
1	377
2	76
3	222

Delen van de standaarddeviaties door het gemiddelde verbruik per dag geeft een percentage van de standaarddeviatie ten opzichte van het verbruik. Deze percentages zet ik in een tabel met de afwijkingspercentages

Afwijkingspercentage	37.45	13.13	45.16	15.04
Standaardafwijkingsperc	47.12	11.35	34.57	18.04

Deze klant heeft een relatief hoog afwijkingspercentage. Maar als je de afwijkingspercentages bekijkt in relatie tot de standaardafwijkingen, dan komen die aardig overeen. Het hoge afwijkingspercentage komt dus door schommelingen in de data, die nou eenmaal niet van tevoren te voorspellen zijn.

Voorbeeld 2:

Tanknummer	0	1	2	3
Gem. afwijking per dag	22.88	256.48	46.35	169.31
Gem. verbruik per dag	63.73	2947.89	229.61	1286.51
Afwijkingspercentage	35.9	8.7	20.19	13.16
Wegingsfactor	4.68	46.02	14.98	34.33

Totaal Gewogen Afwijkingspercentage: 29.54

Standaarddeviaties:

Tank No	St. Dev (liters)
0	18
1	324
2	40
3	178

Afwijkingspercentage	35.90	8.70	20.19	13.16
Standaardafwijkingsperc	28.24	10.99	17.42	13.84

Deze klant heeft een beduidend lager afwijkingspercentage dan de vorige klant. Ook hier het beeld dat het afwijkingspercentage verklaard kan worden vanuit de standaardafwijking.

10.5 Bijlage E: Resultaten van algoritmes op een testset

In onderstaande tabel staan de resultaten van de uitvoering van een performance run op een testset van 11 klanten. Het zijn dus de afwijkingspercentages van het voorspelde verbruik ten opzichte van het werkelijke verbruik.

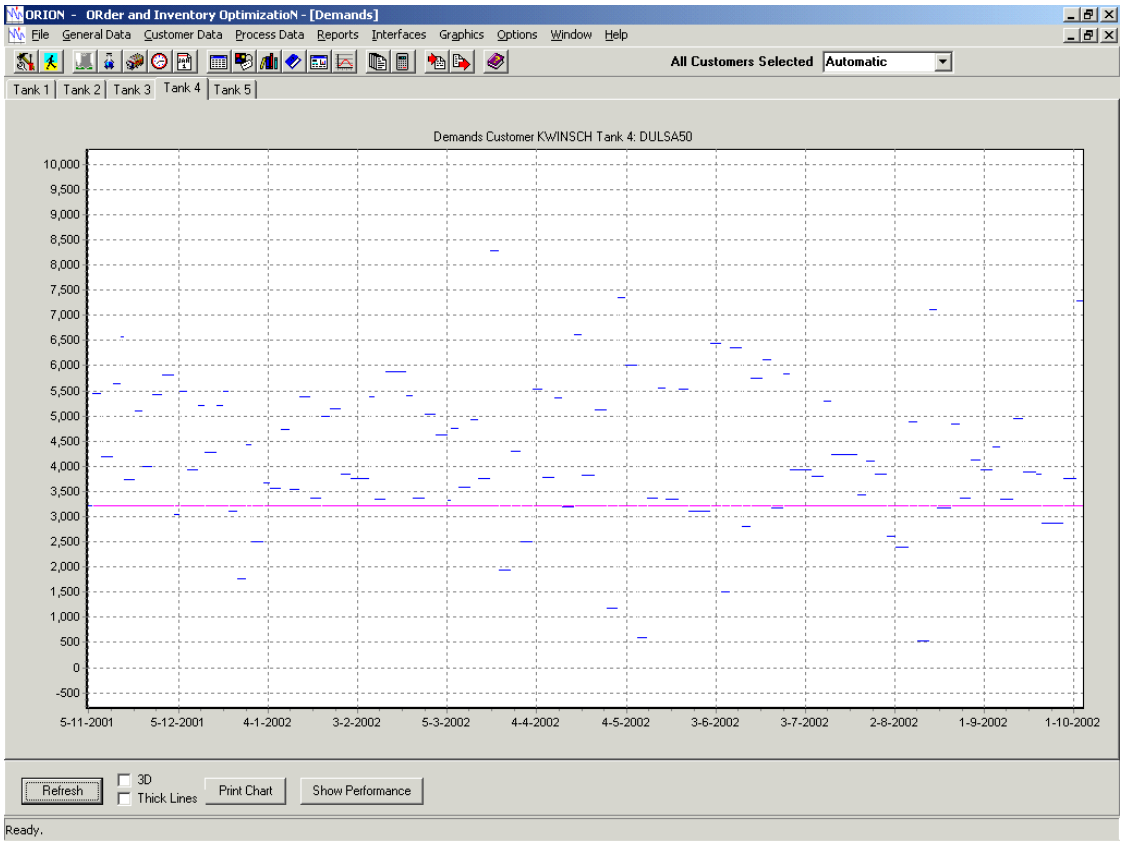
Klant	Afwijking % Mov. Average	Afwijking % Exp. Smoothing
11170	34.86	46.30
11193	29.10	31.63
11201	34.80	54.39
11764	19.25	24.29
12188	24.52	30.11
12196	11.75	15.86
12286	18.59	22.45
12288	37.88	34.24
12550	30.12	32.87
16462	12.24	11.62
16953	18.89	24.04

In 9 van de 11 gevallen geeft ‘moving average’ het beste resultaat. Vaak is het verschil behoorlijk groot. In de 2 gevallen dat ‘exponential smoothing’ beter is, is het verschil tussen beide algoritmes niet zo groot.

Dit resultaat was voor mij aanleiding om eens te kijken naar de instelling van α in het ‘exponential smoothing’ algoritme.

10.6 Bijlage F: Schommelingen in het verbruik

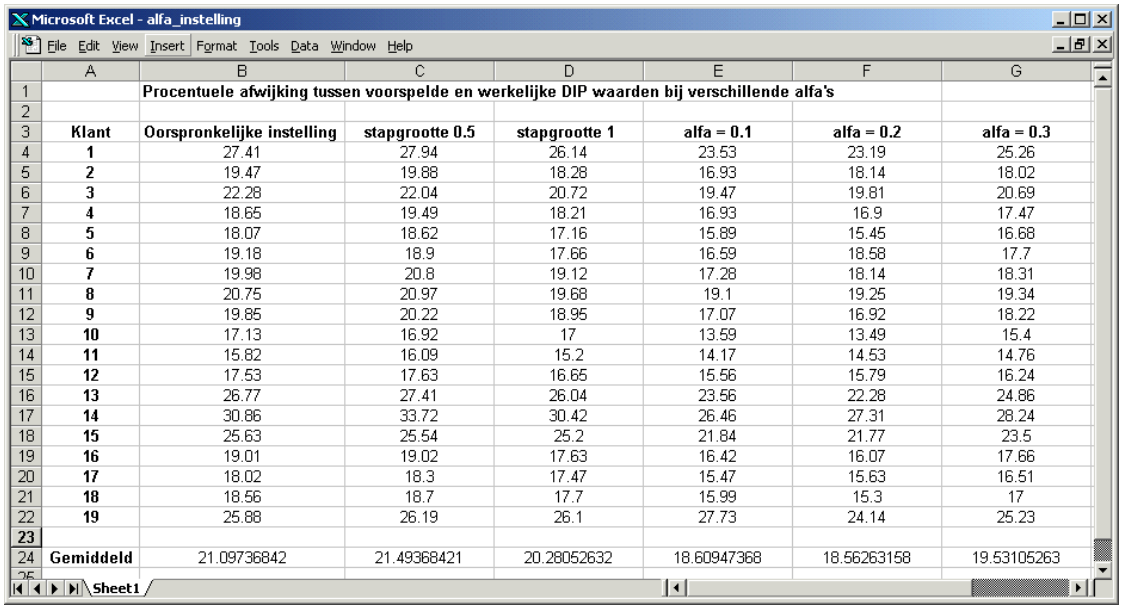
In onderstaand screenshot staan de verbruiksgegevens van een bepaalde klant gedurende een jaar.



De roze lijn geeft het gemiddelde verbruik van de tank aan. (Dit lijkt op het eerste gezicht niet het gemiddelde, vanwege inzoomen op de grafiek). De blauwe streepjes stellen het werkelijke verbruik tussen 2 DIPs voor. Er zijn regelmatig schommelingen in het verbruik. De standaarddeviatie van de waarnemingen is dus hoog.

10.7 Bijlage G: Resultaten bij verschillende waarden voor α

In onderstaande figuur staan de afwijkingspercentages tussen de voorspelde en werkelijke DIPs per klant bij berekening met het ‘exponential smoothing’ algoritme. De kolommen ‘stapgrootte 0.5’ en ‘stapgrootte 1’ corresponderen met ‘instelling 1’ respectievelijk ‘instelling 2’ uit Tabel 5.3.


Microsoft Excel - alfa_instelling

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E	F	G
1		Procentuele afwijking tussen voorspelde en werkelijke DIP waarden bij verschillende alfa's					
2							
3	Klant	Oorspronkelijke instelling	stapgrootte 0.5	stapgrootte 1	alfa = 0.1	alfa = 0.2	alfa = 0.3
4	1	27.41	27.94	26.14	23.53	23.19	25.26
5	2	19.47	19.88	18.28	16.93	18.14	18.02
6	3	22.28	22.04	20.72	19.47	19.81	20.69
7	4	18.65	19.49	18.21	16.93	16.9	17.47
8	5	18.07	18.62	17.16	15.89	15.45	16.68
9	6	19.18	18.9	17.66	16.59	18.58	17.7
10	7	19.98	20.8	19.12	17.28	18.14	18.31
11	8	20.75	20.97	19.68	19.1	19.25	19.34
12	9	19.85	20.22	18.95	17.07	16.92	18.22
13	10	17.13	16.92	17	13.59	13.49	15.4
14	11	15.82	16.09	15.2	14.17	14.53	14.76
15	12	17.53	17.63	16.65	15.56	15.79	16.24
16	13	26.77	27.41	26.04	23.56	22.28	24.86
17	14	30.86	33.72	30.42	26.46	27.31	28.24
18	15	25.63	25.54	25.2	21.84	21.77	23.5
19	16	19.01	19.02	17.63	16.42	16.07	17.66
20	17	18.02	18.3	17.47	15.47	15.63	16.51
21	18	18.56	18.7	17.7	15.99	15.3	17
22	19	25.88	26.19	26.1	27.73	24.14	25.23
23							
24	Gemiddeld	21.09736842	21.49368421	20.28052632	18.60947368	18.56263158	19.53105263

Sheet1

Het blijkt dat de resultaten op deze testset het beste zijn voor α = 0.2. Voor α = 0.1 worden ook goede resultaten gehaald. Ik heb op basis van deze waarnemingen en het feit dat over het algemeen een α tussen de 0.1 en 0.3 de beste resultaten geeft, gekozen voor α = 0.2.